# 目录

入门

架构

简单教程

安装

配置

链接数据源

ODBC/JDBC接口

查询数据

性能调整

日记与调试

SQL参考

数据源和文件格式

开发自定义函数

故障排除

开发者信息

发行说明

# 入门简介

# 架构

# 简单教程

# 安装

## 在分布式模式下启动钻取

2015年12月30日

要以分布模式使用Drill，首先需要在集群中的每个节点上启动Drill守护程序（Drillbit）。尝试连接客户端之前启动Drillbit。要启动钻头，使用**drillbit.sh**命令。

**注意**

如果在嵌入模式下使用钻取，请勿使用drillbit.sh命令。

### 使用drillbit.sh命令

除了启动一个钻头，使用**drillbit.sh**命令来执行其他任务：

* 检查Drillbit的状态
* 停止或重新启动Drillbit
* 配置一个Drillbit默认重启

您可以使用配置文件启动Drillbits。使用这样的文件对于在多个节点上控制Drillbits非常方便。

#### drillbit.sh命令语法

drillbit.sh [--config <conf-dir>] (start|stop|status|restart|autorestart)

例如，要在tarball安装上重新启动Drillbit，请导航到Drill安装目录，然后从安装目录发出以下命令：

bin/drillbit.sh restart

### 启动钻壳

使用Drill shell，您可以使用SQL命令交互式查询连接数据源中的数据。要启动Drill shell，请运行位于Drill安装的bin目录中的以下脚本之一：

* drill-conf  
  使用连接字符串打开Drill shell到in drill-override.conf中指定的ZooKeeper节点<installation directory>/conf。
* drill-localhost  
  使用到本地主机上运行的ZooKeeper的连接打开Drill Shell。

当您启动钻壳时会出现“ [钻”提示](http://drill.apache.org/docs/starting-drill-on-linux-and-mac-os-x/#about-the-drill-prompt)。

发出以下查询以检查在群集中运行的Drillbits：0: jdbc:drill:zk=<zk1host>:<port> SELECT \* FROM sys.drillbits;  
Drill列出有关正在运行的Drillbits的信息，例如主机名和数据端口号。

### 使用Ad-Hoc连接进行钻取

要使用自定义连接进行钻取，但不能修改drill-conf通常使用的连接字符串，请使用以下方式在特定的基础上启动钻取shell sqlline。例如，要使用特定的存储插件作为模式来启动Drill shell，请使用以下命令语法：

sqlline –u jdbc:drill:[schema=<storage plugin>;]zk=<zk name>[:<port>][,<zk name2>[:<port>]... ]

#### sqlline参数和连接参数

* -u是连接字符串之前的选项。需要。
* jdbc是连接类型。需要。
* schema是用作查询默认值的[存储插件](http://drill.apache.org/docs/storage-plugin-registration)配置的名称。可选的。
* zk name 指定一个或多个ZooKeeper主机名或IP地址。
* port是ZooKeeper端口号。可选的。端口2181是Apache Drill的开源版本的默认端口。

例如，使用作为dfs存储插件的默认模式启动Drill shell ：

bin/sqlline –u jdbc:drill:schema=dfs;zk=centos26

在启动时指定存储插件配置名称，不需要USE <schema name>在FROM子句中输入或指定它。

以下命令在配置为在三个节点上运行ZooKeeper的集群中启动Drill shell：

bin/sqlline –u jdbc:drill:zk=cento23,centos24,centos26:5181

### 建立一个直接的钻头连接

如果您想直接连接到一个钻头，而不是使用ZooKeeper的选择钻头，替代zk=<zk name>与drillbit=<node>下面的URL，如图所示。

jdbc:drill:[schema=<storage plugin>;]drillbit=<node name>[:<port>][,<node name2>[:<port>]...  
]<directory>/<cluster ID>

哪里

drillbit=<node name> 指定运行Drill的群集节点的一个或多个主机名或IP地址。

### 退出钻壳

要退出钻壳，请发出以下命令：

!quit

### 停止钻

导航到Drill安装目录，并发出以下命令来停止Drillbit：

bin/drillbit.sh stop

# 配置

# 连接数据源

## 连接数据源简介：

存储插件是Drill用来连接数据源的一个软件模块。存储插件通常优化Drill查询的执行，提供数据的位置，并配置工作区和文件格式以读取数据。Drill安装了多个存储插件，您可以根据自己的环境进行配置。Drill通过存储插件连接到数据源，例如数据库，本地或分布式文件系统上的文件或Hive Metastore。

您可以修改存储插件的默认配置X，并为新配置指定一个唯一的名称Y.本文档将Y称为不同的存储插件，但它实际上只是原始接口的重新配置。在执行查询时，Drill通过以下几种方式之一获取存储插件配置名称：

查询的FROM子句可以标识要使用的插件。

使用 命令可以在查询之前。

您可以在启动Drill取时指定存储插件。

Drill 插件处于apache drill 和data Source 之间。

除了前面提到的功能之外，存储插件还执行扫描程序和写入程序功能，并向执行引擎通知任何本机功能，例如谓词下推，连接和SQL。

## 存储插件注册：

通过存储插件将Drill连接到文件系统，Hive，HBase或其他数据源。在“drill Web控制台”的“存储”选项卡上，您可以查看并重新配置存储插件。如果未启用HTTPS支持（默认），请http://<IP address>:8047/storage转至查看并配置存储插件。IP地址是分布式系统或localhost嵌入式系统中安装的Drillbits之一的主机名或IP地址。如果启用了HTTPS支持，请转到https://<IP address>:8047/storage。

drill安装注册cp，dfs，hbase，hive，和mongo默认存储插件的配置。

cp

指向Drill类路径中的JAR文件，例如employee.json您可以查询的文件。

dfs

指向本地文件系统，但可以将此存储插件配置为指向任何分布式文件系统，例如Hadoop或S3文件系统。

hbase

提供与HBase的连接。

hive

将Drill与Hive文件，HBase和库的元数据抽象相集成，以读取数据并在SerDes和UDF上进行操作。

mongo

提供到MongoDB数据的连接。

在Drill(Mapr-sandbox,这个是某个公司的发行的一个hadoop 版本)沙箱中，dfs存储插件配置将您连接到预先配置了Drill的Hadoop环境。如果您安装了Drill，则将您dfs连接到文件系统的根目录。

注册存储插件配置

要注册新的存储插件配置，请输入存储名称，单击CREATE，以JSON格式提供配置，然后单击UPDATE。

存储插件配置持久性

Drill将存储插件配置保存在临时目录（嵌入模式）或ZooKeeper（分布式模式）中。例如，在Mac OS X上，Drill使用/tmp/drill/sys.storage\_plugins存储插件配置。临时目录在您重新引导时清除。在嵌入式模式下运行钻取时，将sys.store.provider.local.path选项添加到drill-override.conf文件以及用于存储插件配置的路径。例如：

drill.exec: {

cluster-id: "drillbits1",

zk.connect: "localhost:2181",

sys.store.provider.local.path="/mypath"

}

## 存储插件配置（分为如下几个方面）

插件配置基础

工作区

Drill默认输入格式

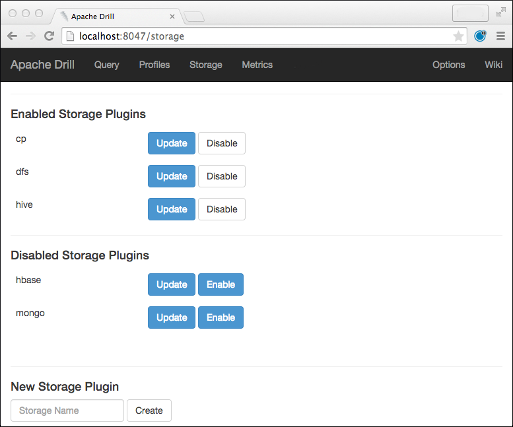
### 插件基础配置

当您在具有多个Drill安装的群集中的一个Drill节点上添加或更新存储插件配置时，Drill会将信息广播到其他Drill节点以同步存储插件配置。添加或更新存储插件配置时，不需要重新启动任何Drillbits。

使用drill Web控制台

你可以使用Dril web控制台更新或添加新的存储插件配置。drillbits需要运行以启动Web控制台。

要创建一个名称和新的配置：

1. [启动钻壳](http://drill.apache.org/docs/starting-drill-on-linux-and-mac-os-x/)。
2. [启动Web控制台](http://drill.apache.org/docs/starting-the-web-console/)。
3. 在存储选项卡上，在**新建存储插件中**输入名称。使用Drill注册的每个配置都必须具有不同的名称。名称区分大小写。  
   

**注意**

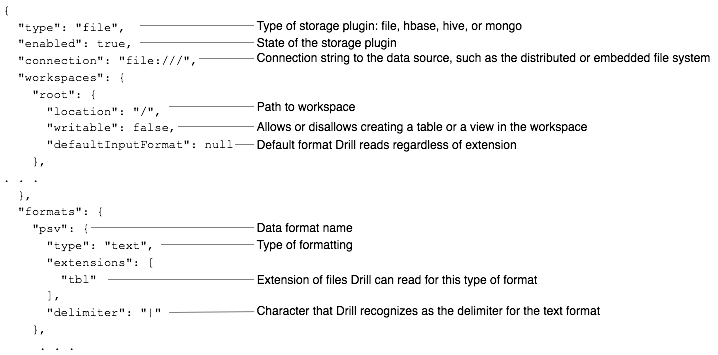
（Web）URL根据您的安装和配置而有所不同。

4，点击**创建**。

5，在配置中，如果可能，请使用JSON格式来修改现有配置的副本。  
使用现有配置的副本可以降低JSON编码错误的风险。使用下一节中的“存储插件属性”表作为进行典型修改的指南。

6，点击**创建**。

存储插件属性

下图显示了典型dfs存储插件配置的关键属性：  


**属性和定义列表**

**List of Attributes and Definitions**

The following table describes the attributes you configure for storage plugins installed with Drill.

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute** | **Example Values** | **Required** | **Description** |
| "type" | "file" "hbase" "hive" "mongo" | yes | A valid storage plugin type name. |
| "enabled" | true false | yes | State of the storage plugin. |
| "connection" | "classpath:///" "file:///" "mongodb://localhost:27017/" "hdfs://" | implementation-dependent | The type of distributed file system, such as HDFS, Amazon S3, or files in your file system, and an address/path name. |
| "workspaces" | null "logs" | no | One or more unique workspace names. If a workspace name is used more than once, only the last definition is effective. |
| "workspaces". . . "location" | "location": "/Users/johndoe/mydata" "location": "/tmp" | no | Full path to a directory on the file system. |
| "workspaces". . . "writable" | true false | no | One or more unique workspace names. If defined more than once, the last workspace name overrides the others. |
| "workspaces". . . "defaultInputFormat" | null "parquet" "csv" "json" | no | Format for reading data, regardless of extension. Default = "parquet" |
| "formats" | "pcap" "psv" "csv" "tsv" "parquet" "json" "avro" "maprdb" "sequencefile" | yes | One or more valid file formats for reading. Drill detects formats of some files; others require configuration. The maprdb format is in installations of the mapr-drill package. |
| "formats" . . . "type" | "pcap" "text" "parquet" "json" "maprdb" "avro" "sequencefile" | yes | Format type. You can define two formats, csv and psv, as type "Text", but having different delimiters. |
| formats . . . "extensions" | ["csv"] | format-dependent | File name extensions that Drill can read. |
| "formats" . . . "delimiter" | "\n" "\r" "\t" "\r\n" "," | format-dependent | Sequence of one or more characters that signifies the end of a line of text and the start of a new line in a delimited text file, such as CSV. Drill treats \n as the standard line delimiter. As of Drill 1.8, Drill supports multi-byte delimiters, such as \r\n. Use a 4-digit hex code syntax \uXXXX for a non-printable delimiter. |
| "formats" . . . "quote" | """ | no | A single character that starts/ends a value in a delimited text file. |
| "formats" . . . "escape" | "`" | no | A single character that escapes a quotation mark inside a value. |
| "formats" . . . "comment" | "#" | no | The line decoration that starts a comment line in the delimited text file. |
| "formats" . . . "skipFirstLine" | true | no | To include or omit the header when reading a delimited text file. Set to true to avoid reading headers as data. |
| "formats" . . . "extractHeader" | true | no | Set to true to extract and use headers as column names when reading a delimited text file, false otherwise. Ensure skipFirstLine is not true when extractHeader=false. |

使用格式属性

您可以formats在存储插件配置的区域中设置格式属性，例如skipFirstLine 。设置文本文件（如CSV）的sys.options属性时，还需要将该属性设置exec.storage.enable\_new\_text\_reader为true（默认值）。有关使用文本文件格式的更多信息和示例，请参阅[“文本文件：CSV，TSV，PSV”](http://drill.apache.org/docs/text-files-csv-tsv-psv/)。

**使用格式属性作为表函数参数**

在Drill版本1.4及更高版本中，您还可以在每个查询的基础上设置上面定义的格式属性。要将参数传递给格式插件，请使用表函数语法：

select a, b from table({table function name}(parameters))

的table function name是表名，类型参数是格式名称，其它参数是所述格式插件配置接受，如表中所定义以上（除字段extensions，其在这种情况下不适用）。

例如，要读取CSV文件并解析标题，请执行以下操作：  
select a, b from table(dfs.`path/to/data.csv`(type => 'text',

fieldDelimiter => ',', extractHeader => true))

有关格式插件配置的更多信息，请参阅[“文本文件：CSV，TSV，PSV”](http://drill.apache.org/docs/text-files-csv-tsv-psv/)。

**使用其他属性**

其它属性，如的配置size.calculator.enabled在hbase插件并configProps在hive插件，是实现相关的，并且超出了本文档的范围。

区分大小写的名称

如前所述，工作区和存储插件名称区分大小写。例如，以下查询使用存储插件名称dfs和工作区名称clicks。当您dfs.clicks在SQL语句中引用时，请使用定义的情况：

0: jdbc:drill:> USE dfs.clicks;

例如，使用小写字母定义存储插件和工作空间名称后，在查询中使用大写字母不起作用。

存储插件REST API

如果您需要将存储插件配置添加到Drill并且不想使用Web浏览器，则可以使用[Drill REST API](http://drill.apache.org/docs/rest-api/#get-status-threads)创建存储插件配置。使用POST请求并传递两个属性：

* 名称  
  存储插件配置名称。
* config   
  在Web控制台中输入的属性设置。

例如，该命令创建一个名为myplugin的存储插件，用于读取位于文件系统根目录下的未知类型的文件：

curl -X POST -H "Content-Type: application/json" -d '{"name":"myplugin", "config": {"type": "file", "enabled": false, "connection": "file:///", "workspaces": { "root": { "location": "/", "writable": false, "defaultInputFormat": null}}, "formats": null}}' http://localhost:8047/storage/myplugin.json

此示例假定HTTPS尚未启用。

**引导存储插件**

建议使用REST API以编程方式向Drill添加存储插件配置。仅在分布式环境中使用的替代方法是自举。您可以创建[bootstrap-storage-plugins.json](https://github.com/apache/drill/blob/master/contrib/storage-hbase/src/main/resources/bootstrap-storage-plugins.json)文件，并在启动Drill时将其包含在类路径中。Drill启动时加载存储插件配置。

目前，引导存储插件配置仅在集群中的第一个Drillbit首次启动时起作用。配置存储在ZooKeeper中，防止Drill再次获取bootstrap-storage-plugins.json。

集群启动后，您必须使用REST API或钻取Web控制台来添加存储插件配置。或者，您可以通过将该插件的json文件上传到zookeeper安装的/ drill目录来修改ZooKeeper中的条目，或者只要删除/ drill目录（如果没有要保留的配置属性）即可修改该条目。

如果使用bootstrap-storage-plugins.json文件加载HBase存储插件配置，并且未安装HBase，则在执行查询时可能会遇到延迟。在HBase插件配置的配置块中配置[HBase客户端超时](http://hbase.apache.org/book.html/#config.files)和重试设置。

### 工作区

您可以在[存储插件配置中](http://drill.apache.org/docs/plugin-configuration-basics/)定义一个或多个工作区。工作空间定义本地或分布式文件系统的子目录中文件的位置。当您运行查询时，Drill将搜索工作区以查找数据。隐藏的默认工作区dfs.default指向文件系统的根目录。

以下dfs存储插件配置显示了定义的工作区的一些示例：

{

"type": "file",

"enabled": true,

"connection": "file:///",

"workspaces": {

"root": {

"location": "/",

"writable": false,

"defaultInputFormat": null

},

"tmp": {

"location": "/tmp",

"writable": true,

"defaultInputFormat": null

},

"emp": {

"location": "/Users/user1/emp",

"writable": true,

"defaultInputFormat": null

},

"donuts": {

"location": "/Users/user1/donuts",

"writable": true,

"defaultInputFormat": null

},

"sales": {

"location": "/Users/user1/sales",

"writable": true,

"defaultInputFormat": null

}

},

配置工作区以包含子目录可简化查询，这在重复查询

配置工作区以包含子目录可简化查询，这在重复查询相同文件时非常重要。在工作空间location属性中配置长路径名称之后，而不是使用数据源的完整路径名称，在FROM子句中使用点符号。

<workspace name>.`<location>`

哪里<location>是一个子目录的路径名称，如[“查询甜甜圈示例”中](http://drill.apache.org/docs/file-system-storage-plugin/#querying-donuts-example)/users/max/drill/json所示的双引号[。](http://drill.apache.org/docs/file-system-storage-plugin/#querying-donuts-example)

要在未将缺省模式名称设置为存储插件配置时查询数据源，请包含插件名称。此语法假定您没有发出USE语句来连接到定义数据位置的存储插件：

<plugin>.<workspace name>.`<location>`

重写 **dfs.default**

您可能希望在用户无权访问根目录的情况下覆盖隐藏的默认工作区。将以下工作空间条目添加到dfs存储插件配置以覆盖默认工作空间：

"default": {

"location": "</directory/path>",

"writable": true,

"defaultInputFormat": null

}

没有工作区的Hive和HBase

虽然Hive数据库在Drill中显示为工作空间，但不能在使用Apache Drill安装的插件hive和hbase插件的配置中包含工作 空间。每个hive存储插件配置都包含一个default指向Hive Metastore 的工作空间。在hive default工作区中查询文件和表格时，可以从查询中省略工作区名称。

例如，您可以default workspace 使用以下任一查询在Hive表上发出查询，并获得相同的结果：

**例**

SELECT \* FROM hive.customers LIMIT 10;

SELECT \* FROM hive.`default`.customers LIMIT 10;

**注意**

默认是保留字。在作为后面标记中的标识符时，必须附上保留字。

由于HBase存储插件不适合工作区，因此可以使用以下查询：

SELECT \* FROM hbase.customers LIMIT 10;

Drill 默认的输入格式：

### 钻取默认输入格式

您可以定义默认的输入格式来告诉Drill在文件系统中的工作区中存在哪种文件类型。

通常，Drill 在搜索工作区时根据文件扩展名和幻数确定文件类型。魔术数字是Drill用来识别Parquet文件的文件签名。如果Drill无法根据文件扩展名或幻数识别文件类型，则查询失败。定义默认的输入格式可以防止查询在Drill无法确定文件类型的情况下失败。

如果您没有在工作区中定义默认的文件类型，或者错误地定义了默认的文件类型，并且Drill无法确定没有此信息的文件类型，则查询失败。您可以为每个工作区定义一个默认的输入格式。您可以为Drill支持的任何文件类型定义默认的输入格式。目前，Drill支持以下输入类型：

* Avro公司
* CSV，TSV或PSV
* 实木复合地板
* JSON
* Hadoop序列文件

您必须先定义一个[工作区，](http://drill.apache.org/docs/workspaces)然后才能定义默认的输入格式。

为工作区定义默认的输入格式：

1. 导航到[钻取Web控制台](http://drill.apache.org/docs/plugin-configuration-basics/#using-the-drill-web-console)。在连接到钻取Web控制台之前，Drillbit进程必须在节点上运行。
2. 在工具栏中选择**存储**。
3. 单击要为其定义工作空间默认输入格式的存储插件配置旁边的**更新**。
4. 在“配置”区域中找到工作区，然后将该defaultInputFormat属性更改为任何受支持的文件类型。

**定义默认输入格式的示例**

{

"type": "file",

"enabled": true,

"connection": "hdfs://",

"workspaces": {

"root": {

"location": "/drill/testdata",

"writable": false,

"defaultInputFormat": "csv"

},

"local" : {

"location" : "/max/proddata",

"writable" : true,

"defaultInputFormat" : "json"

}

## 文件存储插件

### 文件系统存储插件

您可以注册将Drill连接到本地文件系统或Hadoop中注册的分布式文件系统core-site.xml（例如S3或HDFS）的存储插件配置。默认情况下，Apache Drill包含一个存储插件配置dfs，默认情况下，该插件配置指向您计算机上的本地文件系统。

### 将Drill连接到文件系统

在Drill群集中，通常不会查询本地文件系统，而是将文件放在分布式文件系统上。目前，在连接多个Drillbits时，您需要使用分布式文件系统以获取完整一致的查询结果，通过将文件复制到每个节点来模拟分布式文件系统，或者使用Amazon卷（例如Amazon Elastic File System）。

您可以配置存储插件工作空间的连接属性，以将Drill连接到分布式文件系统。例如，以下连接属性将Drill连接到来自客户端的HDFS集群：

"connection": "hdfs://<IP Address>:<Port>/"

要从群集上的节点查询HDFS上的文件，只需将存储插件中的连接更改为file:///即可。hdfs://dfs

要将dfs存储插件配置更改为指向不同的本地或分布式文件系统，请使用connection以下示例中所示的属性。

* 本地文件系统示例：

{

"type": "file",

"enabled": true,

"connection": "file:///",

"workspaces": {

"root": {

"location": "/user/max/donuts",

"writable": false,

"defaultInputFormat": null

}

},

"formats" : {

"json" : {

"type" : "json"

}

}

}

* 分布式文件系统示例：

{

"type" : "file",

"enabled" : true,

"connection" : "hdfs://10.10.30.156:8020/",

"workspaces" : {

"root" : {

"location" : "/user/root/drill",

"writable" : true,

"defaultInputFormat" : null

}

},

"formats" : {

"json" : {

"type" : "json"

}

}

}

要连接到Hadoop文件系统，需要包含名称节点的IP地址和端口号。

### 查询甜甜圈示例

以下示例显示了一个名为的工作空间的文件类型存储插件配置json\_files。配置点钻取到/users/max/drill/json/本地文件系统中的 目录(dfs)：

{

"type" : "file",

"enabled" : true,

"connection" : "file:///",

"workspaces" : {

"json\_files" : {

"location" : "/users/max/drill/json/",

"writable" : false,

"defaultInputFormat" : json

}

},

该connection配置中的参数是“ file:///”，将Drill连接到本地文件系统。

要查询示例json\_files工作区中的文件，可以发出以下USE 命令来告诉Drill使用json\_files工作空间，该工作空间包含在dfs 您发出的每个查询的配置中：(没有试验成功)

USE dfs.json\_files;

SELECT \* FROM `donuts.json` WHERE type='frosted'

如果json\_files工作空间不存在，查询将不得不包含文件的完整文件路径名称donuts.json：

SELECT \* FROM dfs.`/users/max/drill/json/donuts.json` WHERE type='frosted';

## HBase存储插件

当使用安装了Drill的HBase存储插件将Drill连接到HBase数据源时，您需要指定一个ZooKeeper quorum。Drill支持HBase版本0.98。

要查看或更改HBase存储插件配置，请使用[钻取Web控制台](http://drill.apache.org/docs/plugin-configuration-basics/#using-the-drill-web-console)。在Web控制台中，选择**存储**选项卡，然后单击存储插件配置的**更新**按钮hbase。以下示例显示了一个典型的HBase存储插件：

{

"type": "hbase",

"config": {

"hbase.zookeeper.quorum": "10.10.100.62,10.10.10.52,10.10.10.53",

"hbase.zookeeper.property.clientPort": "2181"

},

"size.calculator.enabled": false,

"enabled": true

}

## Hive存储插件

### 2016年1月8日

Drill 1.1和更高版本支持Hive 1.0。要使用自定义SerDes或InputFormat / OutputFormat访问Hive表，所有运行Drillbits的节点必须JAR在<drill\_installation\_directory>/jars/3rdparty文件夹中具有SerDes或InputFormat / OutputFormat 文件 。

您可以按照以下方式通过配置Hive存储插件来运行Hive查询：

* [将Drill连接到Hive远程Metastore](http://drill.apache.org/docs/hive-storage-plugin/#connect-drill-to-the-hive-remote-metastore-directly)
* [连接到Hive嵌入式Metastore](http://drill.apache.org/docs/hive-storage-plugin/#connect-to-the-hive-embedded-metastore)

您通过选择[钻取Web控制台](http://drill.apache.org/docs/plugin-configuration-basics/#using-the-drill-web-console)上的**存储选项卡**来更新Hive存储插件。从Drill Web控制台的已禁用存储插件列表中，单击旁边的**更新**。默认的Hive存储插件配置如下所示：hive

{

"type": "hive",

"enabled": false,

"configProps": {

"hive.metastore.uris": "",

"javax.jdo.option.ConnectionURL": "jdbc:derby:;databaseName=../sample-data/drill\_hive\_db;create=true",

"hive.metastore.warehouse.dir": "/tmp/drill\_hive\_wh",

"fs.default.name": "file:///",

"hive.metastore.sasl.enabled": "false"

}

}

### 将Drill连接到Hive Remote Metastore

Hive Metastore作为Hive之外的单独服务运行。钻子可以通过Thrift查询蜂房Metore。Metastore服务通过JDBC与Hive数据库进行通信。

按照下一节中的步骤将Drill指向Hive Metastore服务地址。在Hive存储插件配置中提供连接参数以配置到Drill的连接。此时，如果您没有使用HBaseStorageHandler查询Hive表，您完成了配置Hive存储插件; 否则，如果使用HBaseStorageHandler查询Hive表，则需要添加ZooKeeper仲裁和端口属性。HBaseStorageHandler需要这些属性。Drill使用这些属性发现HBase服务。如果您使用HBase存储插件，则假设您希望Hive查询访问相同的HBase源，请使用Hive存储插件和HBase存储插件中相同的ZooKeeper仲裁和端口属性。

**注意**

在注册Hive Metastore之前，验证Hive Metastore服务是否正在运行。

#### Hive远程Metastore配置

将Drill连接到远程蜂房Metastore：

1. 发出以下命令以在以下网址中指定的系统上启动Hive Metastore服务hive.metastore.uris：  
   hive --service metastore
2. 在“ [钻取Web控制台”中](http://drill.apache.org/docs/plugin-configuration-basics/#using-the-drill-web-console)，选择“ **存储”**选项卡。
3. 在“钻取Web控制台”中禁用的存储插件列表中，单击**“**旁边的**更新”**hive。
4. 在配置窗口中，添加Thrift URI和端口hive.metastore.uris。例如：
5. ...
6. "configProps": {
7. "hive.metastore.uris": "thrift://<host>:<port>",
8. ...
9. 更改文件的默认位置以适合您的环境; 例如，将"fs.default.name"属性从更改"file:///"为hdfs://或hdfs://<host name>:<port>。在"fs.default.name"包括主机名和端口必须指向主要控制节点。例如：
10. {
11. "type": "hive",
12. "enabled": false,
13. "configProps": {
14. "hive.metastore.uris": "thrift://hdfs41:9083",
15. "hive.metastore.sasl.enabled": "false",
16. "fs.default.name": "hdfs://10.10.10.41/"
17. }
18. }
19. 如果没有查询使用HBaseStorageHandler蜂房表，请跳过此步骤; 否则，请添加ZooKeeper仲裁主机和ZooKeeper端口的名称，例如2181。
20. {
21. "type": "hive",
22. "enabled": false,
23. "configProps": {
24. .
25. .
26. .
27. "hbase.zookeeper.quorum": "zkhost1,zkhost2,zkhost3",
28. "hbase.zookeeper.property.clientPort:" "2181"
29. }
30. }
31. 点击**启用**。

### 连接到Hive嵌入式Metastore

Hive Metastore配置嵌入在Drill过程中。仅在运行单个Drillbit的群集中配置嵌入式元数据，仅用于测试目的。不要将Hive Metastore嵌入到生产系统中。

在“钻取Web控制台”中提供Metastore数据库配置设置。在配置嵌入式Hive Metastore之前，请验证用于连接到Hive Metastore /<drill installation directory>/lib/.的驱动程序是否位于Drill classpath中。如果驱动程序不在该驱动程序中，请将驱动程序复制到/<drill

installation directory>/libDrill节点上。有关存储类型和配置的更多信息，请参阅[“Hive Metastore管理”](https://cwiki.apache.org/confluence/display/Hive/AdminManual+MetastoreAdmin)。

#### Hive嵌入式Metastore配置

要配置嵌入式Hive Metastore，请完成以下步骤：

1. 在“ [钻取Web控制台”中](http://drill.apache.org/docs/plugin-configuration-basics/#using-the-drill-web-console)，选择“ **存储”**选项卡。
2. 在已禁用的存储插件配置部分中，单击旁边的**更新**hive。
3. 在配置窗口中，添加数据库配置设置。

**例**

{

"type": "hive",

"enabled": false,

"configProps": {

"hive.metastore.uris": "",

"javax.jdo.option.ConnectionURL": "jdbc:<database>://<host:port>/<metastore database>",

"hive.metastore.warehouse.dir": "/tmp/drill\_hive\_wh",

"fs.default.name": "file:///",

"hive.metastore.sasl.enabled": "false"

}

}

1. 更改"fs.default.name"属性以指定文件的默认位置。该值需要是一个可用并能够处理文件系统请求的URI。例如，将本地文件系统URI "file:///"更改为HDFS URI：hdfs://或使用namenode将其更改为HDFS 上的路径：hdfs://<hostname>:<port>
2. 点击**启用**。

“javax.jdo.option.ConnectionURL”可能是一个存储元数据的MySQL数据库。Hive Metastore服务提供对像MySQL这样的物理数据库的访问。MySQL存储元数据。

配置Hive存储插件后，可以[查询Hive表](http://drill.apache.org/docs/querying-hive/)。

# 数据查询

查询数据

* [查询数据介绍](http://drill.apache.org/docs/query-data-introduction/)
* [查询文件系统](http://drill.apache.org/docs/querying-a-file-system/)
  + [查询文件系统介绍](http://drill.apache.org/docs/querying-a-file-system-introduction/)
  + [查询Avro文件](http://drill.apache.org/docs/querying-avro-files/)
  + [查询JSON文件](http://drill.apache.org/docs/querying-json-files/)
  + [查询镶木地板文件](http://drill.apache.org/docs/querying-parquet-files/)
  + [查询纯文本文件](http://drill.apache.org/docs/querying-plain-text-files/)
  + [查询目录](http://drill.apache.org/docs/querying-directories/)
  + [查询序列文件](http://drill.apache.org/docs/querying-sequence-files/)
* [查询HBase](http://drill.apache.org/docs/querying-hbase/)
* [查询复杂数据](http://drill.apache.org/docs/querying-complex-data/)
  + [查询复杂数据介绍](http://drill.apache.org/docs/querying-complex-data-introduction/)
  + [示例数据：甜甜圈](http://drill.apache.org/docs/sample-data-donuts/)
  + [选择平面数据](http://drill.apache.org/docs/selecting-flat-data/)
  + [使用SQL函数，子句和联接](http://drill.apache.org/docs/using-sql-functions-clauses-and-joins/)
  + [为列选择嵌套数据](http://drill.apache.org/docs/selecting-nested-data-for-a-column/)
  + [在嵌套数据中选择多列](http://drill.apache.org/docs/selecting-multiple-columns-within-nested-data/)
* [查询蜂巢](http://drill.apache.org/docs/querying-hive/)
* [查询信息模式](http://drill.apache.org/docs/querying-the-information-schema/)
* [查询系统表](http://drill.apache.org/docs/querying-system-tables/)
* [在“钻取Web控制台”中监视和取消查询](http://drill.apache.org/docs/monitoring-and-canceling-queries-in-the-drill-web-console/)

## 查询数据介绍

您可以查询本节的子主题中所述的本地和分布式文件系统，Hive，HBase数据，复杂数据，INFORMATION SCHEMA和系统表。

查询指定数据源位置并包含数据转换。

### 指定数据源位置

可选的[USE语句](http://drill.apache.org/docs/use)针对特定的[存储插件](http://drill.apache.org/docs/connect-a-data-source-introduction/)运行后续查询。USE语句通常保存在FROM语句中输入一些存储插件信息。如果您省略USE语句，请指定一个存储插件（如dfs），以及可选的工作空间（如缺省值）以及使用点符号和反标记的数据源路径。例如：

dfs.`default`.`/Users/drill-user/apache-drill-1.1.0/log/sqlline\_queries.json`;

### 投射数据

在某些情况下，Drill将模式较少的数据隐式转换为正确类型的数据。在这种情况下，您不需要[将数据转换](http://drill.apache.org/docs/supported-data-types/#casting-and-converting-data-types)为其他类型。数据的文件格式和查询的性质决定了转换或转换的需求。铸造中的差异取决于数据源。

例如，您必须将"100"JSON文件中的字符串转换为整数才能应用数学函数或聚合函数。

要使用Drill查询HBase数据，请按照[“查询HBase”](http://drill.apache.org/docs/querying-hbase/)一节所述，将HBase表的每一列转换为/从字节数组转换为SQL数据类型。使用[CONVERT\_TO或CONVERT\_FROM](http://drill.apache.org/docs/data-type-conversion/#convert_to-and-convert_from)函数来执行HBase数据的转换。

### 疑难解答查询

除了在Drill Shell中以交互方式测试查询，并检查错误消息之外，还可以使用[EXPLAIN命令](http://drill.apache.org/docs/explain/)分析错误并对不运行的查询进行疑难解答。例如，如果遇到转换错误，则查询计划文本可能会帮助您隔离问题。

0: jdbc:drill:zk=local> !set maxwidth 10000

0: jdbc:drill:zk=local> explain plan for select ... ;

[钻壳命令](http://drill.apache.org/docs/configuring-the-drill-shell/)包括!set <set variable> <value>增加默认的文本显示（字符数）。默认情况下，大多数计划输出是隐藏的。

### 查询语法提示

使用Drill查询数据时请记住以下提示：

* 在SQL语句的末尾添加分号，除非发出[Drill shell命令](http://drill.apache.org/docs/configuring-the-drill-shell/)。  
  Example:！set maxwidth 10000`
* 在[关键字](http://drill.apache.org/docs/reserved-keywords)，特殊字符以及SQL无法解析的[标识符](http://drill.apache.org/docs/lexical-structure/#identifier)（如关键字default和包含正斜杠字符的路径）中使用反引号：示例：SELECT \* FROM dfs.`default`.`/Users/drilluser/apache-drill-1.1.0/sample-data/nation.parquet`;
* 选择全部（SELECT \*）无模式数据时，返回列的顺序可能与存储的顺序不同，并且可能因查询而异。

## 查询文件系统

* [查询文件系统介绍](http://drill.apache.org/docs/querying-a-file-system-introduction/)
* [查询Avro文件](http://drill.apache.org/docs/querying-avro-files/)
* [查询JSON文件](http://drill.apache.org/docs/querying-json-files/)
* [查询镶木地板文件](http://drill.apache.org/docs/querying-parquet-files/)
* [查询纯文本文件](http://drill.apache.org/docs/querying-plain-text-files/)
* [查询目录](http://drill.apache.org/docs/querying-directories/)
* [查询序列文件](http://drill.apache.org/docs/querying-sequence-files/)

### 查询镶木地板文件

2016年11月21日

Drill安装包括一个sample-data可以查询的具有Parquet文件的目录。使用SQL查询region.parquet和nation.parquet文件的sample-data目录。

**注意**

钻孔安装位置可能与此处使用的示例不同。

这些例子假设Drill是以[嵌入模式安装的](http://drill.apache.org/docs/installing-drill-in-embedded-mode)。如果您以分布模式安装Drill，或者您的sample-data目录与示例中使用的位置不同。sample-data在运行查询之前将目录更改为正确的位置。

#### 区域文件

要查看region.parquet文件中的数据，请发出以下查询：

SELECT \* FROM dfs.`<path-to-installation>/apache-drill-<version>/sample-data/region.parquet`;

该查询返回以下结果：

+--------------+--------------+-----------------------+

| R\_REGIONKEY | R\_NAME | R\_COMMENT |

+--------------+--------------+-----------------------+

| 0 | AFRICA | lar deposits. blithe |

| 1 | AMERICA | hs use ironic, even |

| 2 | ASIA | ges. thinly even pin |

| 3 | EUROPE | ly final courts cajo |

| 4 | MIDDLE EAST | uickly special accou |

+--------------+--------------+-----------------------+

5 rows selected (0.272 seconds)

#### 民族文件

如果您按照10分钟内的Apache Drill说明在嵌入模式下安装Drill，则不同操作系统之间的parquet文件路径会有所不同。

要查看nation.parquet文件中的数据，请发出适合您的操作系统的查询：

SELECT \* FROM dfs.`<path-to-installation>/sample-data/nation.parquet`;

该查询返回以下结果：

+--------------+-----------------+--------------+-----------------------+

| N\_NATIONKEY | N\_NAME | N\_REGIONKEY | N\_COMMENT |

+--------------+-----------------+--------------+-----------------------+

| 0 | ALGERIA | 0 | haggle. carefully f |

| 1 | ARGENTINA | 1 | al foxes promise sly |

| 2 | BRAZIL | 1 | y alongside of the p |

| 3 | CANADA | 1 | eas hang ironic, sil |

| 4 | EGYPT | 4 | y above the carefull |

| 5 | ETHIOPIA | 0 | ven packages wake qu |

| 6 | FRANCE | 3 | refully final reques |

| 7 | GERMANY | 3 | l platelets. regular |

| 8 | INDIA | 2 | ss excuses cajole sl |

| 9 | INDONESIA | 2 | slyly express asymp |

| 10 | IRAN | 4 | efully alongside of |

| 11 | IRAQ | 4 | nic deposits boost a |

| 12 | JAPAN | 2 | ously. final, expres |

| 13 | JORDAN | 4 | ic deposits are blit |

| 14 | KENYA | 0 | pending excuses hag |

| 15 | MOROCCO | 0 | rns. blithely bold c |

| 16 | MOZAMBIQUE | 0 | s. ironic, unusual a |

| 17 | PERU | 1 | platelets. blithely |

| 18 | CHINA | 2 | c dependencies. furi |

| 19 | ROMANIA | 3 | ular asymptotes are |

| 20 | SAUDI ARABIA | 4 | ts. silent requests |

| 21 | VIETNAM | 2 | hely enticingly expr |

| 22 | RUSSIA | 3 | requests against th |

| 23 | UNITED KINGDOM | 3 | eans boost carefully |

| 24 | UNITED STATES | 1 | y final packages. sl |

+--------------+-----------------+--------------+-----------------------+

25 rows selected (0.102 seconds)

### 查询目录

2016年11月21日

您可以将多个文件存储在一个目录中，并将它们查询为单个实体。您不必显式加入文件。这些文件必须是兼容的，因为它们必须具有相同的数据类型和列。没有可比数据类型的隐藏文件可能会导致[表未找到](http://drill.apache.org/docs/troubleshooting/#table-not-found)错误。您可以查询具有Drill支持格式的文件的目录，例如JSON，Parquet或文本文件。

例如，假设一个testdata目录包含两个具有相同结构的文件：plays.csv和moreplays.csv。第一个文件包含7个记录，第二个文件包含3个记录。以下查询返回由第一列排序的两个文件的“联合”：

0: jdbc:drill:zk=local> SELECT COLUMNS[0] AS `Year`, COLUMNS[1] AS Play

FROM dfs.`/Users/brumsby/drill/testdata` order by 1;

+------------+------------------------+

| Year | Play |

+------------+------------------------+

| 1594 | Comedy of Errors |

| 1595 | Romeo and Juliet |

| 1596 | The Merchant of Venice |

| 1599 | As You Like It |

| 1599 | Hamlet |

| 1601 | Twelfth Night |

| 1606 | Macbeth |

| 1606 | King Lear |

| 1609 | The Winter's Tale |

| 1610 | The Tempest |

+------------+------------------------+

10 rows selected (0.296 seconds)

您可以深入查看并自动查询子目录。例如，假设您有一个日志目录，其中包含每个年份的子目录和每个月份（1到12）的子目录。月份目录包含JSON文件。

[root@ip-172-16-1-200 logs]# pwd

/mapr/drilldemo/labs/clicks/logs

[root@ip-172-16-1-200 logs]# ls

2012 2013 2014

[root@ip-172-16-1-200 logs]# cd 2013

[root@ip-172-16-1-200 2013]# ls

1 10 11 12 2 3 4 5 6 7 8 9

您可以通过在查询中引用文件系统来查询所有这些文件或子集。例如，以下查询计算目录中所有文件中的记录数2013：

0: jdbc:drill:> SELECT COUNT(\*) FROM MFS.`/mapr/drilldemo/labs/clicks/logs/2013` ;

+------------+

| EXPR$0 |

+------------+

| 24000 |

+------------+

1 row selected (2.607 seconds)

您还可以使用变量dir0，dir1等，指的子目录在您的工作空间路径。例如，假定bob.logdata 是一个指向工作空间logs目录，它包含多个子目录：2012，2013，和2014。以下查询约束名为的子目录内的文件2013。变量dir0是指从日志dir1到下一个级别的第一个级别，依此类推。

0: jdbc:drill:> USE bob.logdata;

+------------+-----------------------------------------+

| ok | summary |

+------------+-----------------------------------------+

| true | Default schema changed to 'bob.logdata' |

+------------+-----------------------------------------+

1 row selected (0.305 seconds)

0: jdbc:drill:> SELECT \* FROM logs WHERE dir0='2013' LIMIT 10;

+------------+------------+------------+------------+------------+------------+------------+------------+------------+-------------+

| dir0 | dir1 | trans\_id | date | time | cust\_id | device | state | camp\_id | keywords |

+------------+------------+------------+------------+------------+------------+------------+------------+------------+-------------+

| 2013 | 2 | 12115 | 02/23/2013 | 19:48:24 | 3 | IOS5 | az | 5 | who's |

| 2013 | 2 | 12127 | 02/26/2013 | 19:42:03 | 11459 | IOS5 | wa | 10 | for |

| 2013 | 2 | 12138 | 02/09/2013 | 05:49:01 | 1 | IOS6 | ca | 7 | minutes |

| 2013 | 2 | 12139 | 02/23/2013 | 06:58:20 | 1 | AOS4.4 | ms | 7 | i |

| 2013 | 2 | 12145 | 02/10/2013 | 10:14:56 | 10 | IOS5 | mi | 6 | wrong |

| 2013 | 2 | 12157 | 02/15/2013 | 02:49:22 | 102 | IOS5 | ny | 5 | want |

| 2013 | 2 | 12176 | 02/19/2013 | 08:39:02 | 28 | IOS5 | or | 0 | and |

| 2013 | 2 | 12194 | 02/24/2013 | 08:26:17 | 125445 | IOS5 | ar | 0 | say |

| 2013 | 2 | 12236 | 02/05/2013 | 01:40:05 | 10 | IOS5 | nj | 2 | sir |

| 2013 | 2 | 12249 | 02/03/2013 | 04:45:47 | 21725 | IOS5 | nj | 5 | no |

+------------+------------+------------+------------+------------+------------+------------+------------+------------+-------------+

10 rows selected (0.583 seconds)

## 查询信息模式

2017年4月5日

当您使用Drill连接到多个数据源时，您需要一个简单的机制来发现每个数据源包含的内容。信息模式是ANSI标准的元数据表集，您可以查询这些元数据表以返回有关所有Drill数据源（或模式）的信息。数据源可能是数据库或文件系统; 在这方面他们都被称为“模式”。您可以查询以下INFORMATION\_SCHEMA表：

* SCHEMATA
* CATALOGS
* TABLES
* 列
* VIEWS

### SCHEMATA

SCHEMATA表包含CATALOG\_NAME和SCHEMA\_NAME列。为了在BI工具中提供最大的灵活性，Drill支持的唯一目录是 DRILL。

SELECT CATALOG\_NAME, SCHEMA\_NAME as all\_my\_data\_sources FROM INFORMATION\_SCHEMA.SCHEMATA ORDER BY SCHEMA\_NAME;

+--------------+---------------------+

| CATALOG\_NAME | all\_my\_data\_sources |

+--------------+---------------------+

| DRILL | INFORMATION\_SCHEMA |

| DRILL | cp.default |

| DRILL | dfs.default |

| DRILL | dfs.root |

| DRILL | dfs.tmp |

| DRILL | HiveTest.SalesDB |

| DRILL | maprfs.logs |

| DRILL | sys |

+--------------+---------------------+

INFORMATION\_SCHEMA名称和关联的关键字区分大小写。您还可以通过运行SHOW DATABASES命令来返回模式列表：

SHOW DATABASES;

+-------------+

| SCHEMA\_NAME |

+-------------+

| dfs.default |

| dfs.root |

| dfs.tmp |

...

### CATALOGS

CATALOGS表只返回一行，硬编码的DRILL目录名称和描述。

### TABLES

TABLES表返回数据库中每个表或视图的表名和类型。（类型表示TABLE或VIEW。）请注意，Drill不会返回可用于在基于文件的数据源中查询的文件。而是使用SHOW FILES来浏览这些数据源。

### 列

COLUMNS表返回每个表或视图中每列的列名称和其他元数据（如数据类型）。

### VIEWS

VIEWS表返回数据库中每个视图的名称和定义。请注意，文件模式是Drill中视图的规范存储库。根据您创建视图的方式，只能在使用后才显示在“钻取”中。

### 有用的查询

运行INFORMATION\_SCHEMA.`TABLES`查询以查看数据库中的所有表和视图。TABLES是Drill中的一个保留字，需要反标（`）。

例如，以下查询标识Drill可以访问的所有表和视图：

SELECT TABLE\_SCHEMA, TABLE\_NAME, TABLE\_TYPE

FROM INFORMATION\_SCHEMA.`TABLES`

ORDER BY TABLE\_NAME DESC;

----------------------------------------------------------------

TABLE\_SCHEMA TABLE\_NAME TABLE\_TYPE

----------------------------------------------------------------

HiveTest.CustomersDB Customers TABLE

HiveTest.SalesDB Orders TABLE

HiveTest.SalesDB OrderLines TABLE

HiveTest.SalesDB USOrders VIEW

dfs.default CustomerSocialProfile VIEW

----------------------------------------------------------------

**注意**

目前，Drill只支持查询Drill视图; Hive视图还不被支持。

您可以运行一个类似的查询来标识表中的列以及这些列的数据类型：

SELECT COLUMN\_NAME, DATA\_TYPE

FROM INFORMATION\_SCHEMA.COLUMNS

WHERE TABLE\_NAME = 'Orders' AND TABLE\_SCHEMA = 'HiveTest.SalesDB' AND COLUMN\_NAME LIKE '%Total';

+-------------+------------+

| COLUMN\_NAME | DATA\_TYPE |

+-------------+------------+

| OrderTotal | Decimal |

+-------------+------------+

在此版本中，Drill将禁用DECIMAL数据类型，包括转换为DECIMAL并从Parquet和Hive读取DECIMAL类型。您可以[启用DECIMAL类型](http://drill.apache.org/docs/supported-data-types/#enabling-the-decimal-type)，但不建议这样做。

## 查询HBase

2016年7月22日

本节涵盖以下主题：

* [教程 - 查询HBase数据](http://drill.apache.org/docs/querying-hbase/#tutorial-querying-hbase-data)  
  一个简单的教程，演示如何使用Drill查询HBase数据。
* [使用HBase字节数组](http://drill.apache.org/docs/querying-hbase/#working-with-hbase-byte-arrays)  
  如何使用HBase字节数组来处理严重的应用程序
* [查询Big Endian编码数据](http://drill.apache.org/docs/querying-hbase/#querying-big-endian-encoded-data)  
  如何使用Drill 1.2及更高版本中的优化功能
* [利用HBase有序字节编码](http://drill.apache.org/docs/querying-hbase/#leveraging-hbase-ordered-byte-encoding)  
  如何使用Drill 1.2来利用[HBASE-8201 Jira](https://issues.apache.org/jira/browse/HBASE-8201)引入的新功能

### 教程 - 查询HBase数据

本教程将演示如何将Drill连接到HBase数据源，创建简单的HBase表以及使用Drill查询数据。

#### 配置HBase存储插件

要使用Drill查询HBase数据源，请首先为您的环境配置[HBase存储插件](http://drill.apache.org/docs/hbase-storage-plugin/)。

#### 创建HBase表

您可以在HBase中创建两个表格，学生和点击，您可以使用Drill进行查询。您可以使用CONVERT\_TO和CONVERT\_FROM函数将二进制文本转换为类型化数据。在[查询HBase表的](http://drill.apache.org/docs/querying-hbase/#query-hbase-tables)第4步中，使用CAST函数将二进制数据转换为INT 。在转换INT或BIGINT号码时，如果目标/源中的字节数与二进制源/目标中的数字字节数不匹配，请使用CAST。

要创建HBase表，请完成以下步骤：

1. 将以下命令传递给HBase shell来创建学生并单击HBase中的表：
2. echo "create 'students','account','address'" | hbase shell
3. echo "create 'clicks','clickinfo','iteminfo'" | hbase shell
4. 发出以下命令来创建一个testdata.txt文件：

cat > testdata.txt

1. 将以下put命令复制并粘贴到**cat**命令下面的行中。按Return键，然后按CTRL Z关闭文件。
2. put 'students','student1','account:name','Alice'
3. put 'students','student1','address:street','123 Ballmer Av'
4. put 'students','student1','address:zipcode','12345'
5. put 'students','student1','address:state','CA'
6. put 'students','student2','account:name','Bob'
7. put 'students','student2','address:street','1 Infinite Loop'
8. put 'students','student2','address:zipcode','12345'
9. put 'students','student2','address:state','CA'
10. put 'students','student3','account:name','Frank'
11. put 'students','student3','address:street','435 Walker Ct'
12. put 'students','student3','address:zipcode','12345'
13. put 'students','student3','address:state','CA'
14. put 'students','student4','account:name','Mary'
15. put 'students','student4','address:street','56 Southern Pkwy'
16. put 'students','student4','address:zipcode','12345'
17. put 'students','student4','address:state','CA'
18. put 'clicks','click1','clickinfo:studentid','student1'
19. put 'clicks','click1','clickinfo:url','http://www.google.com'
20. put 'clicks','click1','clickinfo:time','2014-01-01 12:01:01.0001'
21. put 'clicks','click1','iteminfo:itemtype','image'
22. put 'clicks','click1','iteminfo:quantity','1'
23. put 'clicks','click2','clickinfo:studentid','student1'
24. put 'clicks','click2','clickinfo:url','http://www.amazon.com'
25. put 'clicks','click2','clickinfo:time','2014-01-01 01:01:01.0001'
26. put 'clicks','click2','iteminfo:itemtype','image'
27. put 'clicks','click2','iteminfo:quantity','1'
28. put 'clicks','click3','clickinfo:studentid','student2'
29. put 'clicks','click3','clickinfo:url','http://www.google.com'
30. put 'clicks','click3','clickinfo:time','2014-01-01 01:02:01.0001'
31. put 'clicks','click3','iteminfo:itemtype','text'
32. put 'clicks','click3','iteminfo:quantity','2'
33. put 'clicks','click4','clickinfo:studentid','student2'
34. put 'clicks','click4','clickinfo:url','http://www.ask.com'
35. put 'clicks','click4','clickinfo:time','2013-02-01 12:01:01.0001'
36. put 'clicks','click4','iteminfo:itemtype','text'
37. put 'clicks','click4','iteminfo:quantity','5'
38. put 'clicks','click5','clickinfo:studentid','student2'
39. put 'clicks','click5','clickinfo:url','http://www.reuters.com'
40. put 'clicks','click5','clickinfo:time','2013-02-01 12:01:01.0001'
41. put 'clicks','click5','iteminfo:itemtype','text'
42. put 'clicks','click5','iteminfo:quantity','100'
43. put 'clicks','click6','clickinfo:studentid','student3'
44. put 'clicks','click6','clickinfo:url','http://www.google.com'
45. put 'clicks','click6','clickinfo:time','2013-02-01 12:01:01.0001'
46. put 'clicks','click6','iteminfo:itemtype','image'
47. put 'clicks','click6','iteminfo:quantity','1'
48. put 'clicks','click7','clickinfo:studentid','student3'
49. put 'clicks','click7','clickinfo:url','http://www.ask.com'
50. put 'clicks','click7','clickinfo:time','2013-02-01 12:45:01.0001'
51. put 'clicks','click7','iteminfo:itemtype','image'
52. put 'clicks','click7','iteminfo:quantity','10'
53. put 'clicks','click8','clickinfo:studentid','student4'
54. put 'clicks','click8','clickinfo:url','http://www.amazon.com'
55. put 'clicks','click8','clickinfo:time','2013-02-01 22:01:01.0001'
56. put 'clicks','click8','iteminfo:itemtype','image'
57. put 'clicks','click8','iteminfo:quantity','1'
58. put 'clicks','click9','clickinfo:studentid','student4'
59. put 'clicks','click9','clickinfo:url','http://www.amazon.com'
60. put 'clicks','click9','clickinfo:time','2013-02-01 22:01:01.0001'
61. put 'clicks','click9','iteminfo:itemtype','image'
62. put 'clicks','click9','iteminfo:quantity','10'
63. 发出以下命令将数据放入HBase：
64. cat testdata.txt | hbase shell

#### 查询HBase表

[启动Drill](http://drill.apache.org/docs/installing-drill-in-embedded-mode/)并完成以下步骤来查询您创建的HBase表。

1. 使用HBase存储插件配置。  
   USE hbase;
2. 发出以下查询以查看学生表中的数据：  
   SELECT \* FROM students;

查询返回不可用的结果。在下一步中，将数据从字节数组转换为有意义的UTF8类型。

+-------------+-----------------------+---------------------------------------------------------------------------+

| row\_key | account | address |

+-------------+-----------------------+---------------------------------------------------------------------------+

| [B@e6d9eb7 | {"name":"QWxpY2U="} | {"state":"Q0E=","street":"MTIzIEJhbGxtZXIgQXY=","zipcode":"MTIzNDU="} |

| [B@2823a2b4 | {"name":"Qm9i"} | {"state":"Q0E=","street":"MSBJbmZpbml0ZSBMb29w","zipcode":"MTIzNDU="} |

| [B@3b8eec02 | {"name":"RnJhbms="} | {"state":"Q0E=","street":"NDM1IFdhbGtlciBDdA==","zipcode":"MTIzNDU="} |

| [B@242895da | {"name":"TWFyeQ=="} | {"state":"Q0E=","street":"NTYgU291dGhlcm4gUGt3eQ==","zipcode":"MTIzNDU="} |

+-------------+-----------------------+---------------------------------------------------------------------------+

4 rows selected (1.335 seconds)

1. 发出以下查询，其中包括CONVERT\_FROM函数，将students表转换为键入的数据：
2. SELECT CONVERT\_FROM(row\_key, 'UTF8') AS studentid,
3. CONVERT\_FROM(students.account.name, 'UTF8') AS name,
4. CONVERT\_FROM(students.address.state, 'UTF8') AS state,
5. CONVERT\_FROM(students.address.street, 'UTF8') AS street,
6. CONVERT\_FROM(students.address.zipcode, 'UTF8') AS zipcode
7. FROM students;

**注意**

使用点表示法向下钻取到HBase表中的列：tablename.columnfamilyname.columnnname

查询返回的结果看起来好多了：

+------------+------------+------------+------------------+------------+

| studentid | name | state | street | zipcode |

+------------+------------+------------+------------------+------------+

| student1 | Alice | CA | 123 Ballmer Av | 12345 |

| student2 | Bob | CA | 1 Infinite Loop | 12345 |

| student3 | Frank | CA | 435 Walker Ct | 12345 |

| student4 | Mary | CA | 56 Southern Pkwy | 12345 |

+------------+------------+------------+------------------+------------+

4 rows selected (0.504 seconds)

1. 查询点击表以查看哪些学生访问了google.com：
2. SELECT CONVERT\_FROM(row\_key, 'UTF8') AS clickid,
3. CONVERT\_FROM(clicks.clickinfo.studentid, 'UTF8') AS studentid,
4. CONVERT\_FROM(clicks.clickinfo.`time`, 'UTF8') AS `time`,
5. CONVERT\_FROM(clicks.clickinfo.url, 'UTF8') AS url
6. FROM clicks WHERE clicks.clickinfo.url LIKE '%google%';
7. +------------+------------+--------------------------+-----------------------+
8. | clickid | studentid | time | url |
9. +------------+------------+--------------------------+-----------------------+
10. | click1 | student1 | 2014-01-01 12:01:01.0001 | http://www.google.com |
11. | click3 | student2 | 2014-01-01 01:02:01.0001 | http://www.google.com |
12. | click6 | student3 | 2013-02-01 12:01:01.0001 | http://www.google.com |
13. +------------+------------+--------------------------+-----------------------+
14. 3 rows selected (0.294 seconds)
15. 查询点击表以获得具有100个项目的学生的studentid。使用CONVERT\_FROM转换文本studentid和itemtype数据，但使用CAST转换整数量。
16. SELECT CONVERT\_FROM(tbl.clickinfo.studentid, 'UTF8') AS studentid,
17. CONVERT\_FROM(tbl.iteminfo.itemtype, 'UTF8'),
18. CAST(tbl.iteminfo.quantity AS INT) AS items
19. FROM clicks tbl WHERE tbl.iteminfo.quantity=100;
20. +------------+------------+------------+
21. | studentid | EXPR$1 | items |
22. +------------+------------+------------+
23. | student2 | text | 100 |
24. +------------+------------+------------+
25. 1 row selected (0.656 seconds)

#### 使用HBase字节数组

上一节中的小例子查询HBase中的小端编码数据。对于严重的应用程序，您需要了解如何使用HBase字节数组。如果您想让Drill将底层HBase行键解释为除字节数组之外的其他字段，则需要知道HBase中数据的编码。默认情况下，HBase以小端存储数据，Drill认为数据是小端，这是未排序的。下表显示了典型的行密钥ID（以字节为单位）的排序，分别以little endian和big endian编码：

| **字节表示中的ID Little Endian排序** | **十进制符号中的ID** | **字节表示法中的ID Big Endian排序** | **十进制符号中的ID** |
| --- | --- | --- | --- |
| 0 x 010000。。。000 | 1 | 0 x 00000001 | 1 |
| 0×010100。。。000 | 17 | 0 x 00000002 | 2 |
| 0 x 020000。。。000 | 2 | 0 x 00000003 | 3 |
| 。。。 |  | 0 x 00000004 | 4 |
| 0 x 050000。。。000 | 五 | 0 x 00000005 | 五 |
| 。。。 |  | 。。。 |  |
| 0×0A0000。。。000 | 10 | 0×0000000A | 10 |
|  |  | 0 x 00000101 | 17 |

#### 查询大端编码的数据

当您在大端编码数据上使用[“CONVERT\_TO和CONVERT\_FROM数据类型”](http://drill.apache.org/docs/supported-data-types/#data-types-for-convert_to-and-convert_from-functions)时，Drill优化了HBase表的扫描。Drill提供用于CONVERT\_TO和CONVERT\_FROM的\* \_BE编码类型以利用这些优化。以下是一些\* \_BE类型的例子。

* DATE\_EPOCH\_BE
* TIME\_EPOCH\_BE
* TIMESTAMP\_EPOCH\_BE
* UINT8\_BE
* BIGINT\_BE

例如，当您在大端编码数据上使用以下查询时，Drill会以一定的方式返回结果：

SELECT

CONVERT\_FROM(BYTE\_SUBSTR(row\_key, 1, 8), 'DATE\_EPOCH\_BE') d,

CONVERT\_FROM(BYTE\_SUBSTR(row\_key, 9, 8), 'BIGINT\_BE') id,

CONVERT\_FROM(tableName.f.c, 'UTF8')

FROM hbase.`TestTableCompositeDate` tableName

WHERE

CONVERT\_FROM(BYTE\_SUBSTR(row\_key, 1, 8), 'DATE\_EPOCH\_BE') < DATE '2015-06-18' AND

CONVERT\_FROM(BYTE\_SUBSTR(row\_key, 1, 8), 'DATE\_EPOCH\_BE') > DATE '2015-06-13';

此查询假定表的行键表示以big-endian格式编码的DATE\_EPOCH类型。Drill HBase插件将能够修剪扫描范围，因为行键的大端编码前缀存在条件。有关更多示例，请参阅[测试代码](https://github.com/apache/drill/blob/95623912ebf348962fe8a8846c5f47c5fdcf2f78/contrib/storage-hbase/src/test/java/org/apache/drill/hbase/TestHBaseFilterPushDown.java)。

查询HBase数据：

1. 使用[HBase存储插件](http://drill.apache.org/docs/hbase-storage-plugin/)将数据源连接到Drill 。

USE hbase;

1. 确定要查询的HBase数据的编码。询问负责创建数据的人员。
2. 根据数据的编码类型，在查询数据时，使用[“CONVERT\_TO和CONVERT\_FROM数据类型”](http://drill.apache.org/docs/supported-data-types/#data-types-for-convert_to-and-convert_from-functions)将HBase二进制表示转换为SQL类型。  
   例如，在您的Drill查询中使用CONVERT\_FROM将大端编码的行键转换为SQL BIGINT类型：

SELECT CONVERT\_FROM(BYTE\_SUBSTR(row\_key, 1, 8),'BIGINT\_BE’) FROM my\_hbase\_table;

在这个例子中，[BYTE\_SUBSTR函数](http://drill.apache.org/docs/string-manipulation/#byte_substr)分离HBase复合键的各个部分。Drill优化基于Drill 1.2和更高版本的功能，当HBase数据采用big endian格式时，将条件筛选器推送到存储层。

如果只有组合键的第一个组件以big endian格式进行编码，那么Drill可以正常查询使用组合键的HBase数据，如上例所示。如果HBase行键没有以big endian存储，请不要使用\* \_BE类型。如果要将小端字节数组转换为整数，请使用BIGINT而不是BIGINT\_BE，例如，作为CONVERT\_FROM的参数。

#### 利用HBase有序字节编码

Drill 1.2利用[HBASE-8201 Jira](https://issues.apache.org/jira/browse/HBASE-8201)引入的新功能，允许对不同数据类型进行有序字节编码。当数据作为排序字节数组存储在磁盘上时，此编码方案保留本机数据类型的排序顺序。因此，如果行键已经以OrderedBytes格式编码，Drill将能够通过HBase存储插件处理数据。

为了执行下面的查询，Drill修剪扫描范围，只包含代表[-32,59]范围的行键，从而减少读取的数据量。

SELECT

CONVERT\_FROM(t.row\_key, 'INT\_OB') rk,

CONVERT\_FROM(t.`f`.`c`, 'UTF8') val

FROM

hbase.`TestTableIntOB` t

WHERE

CONVERT\_FROM(row\_key, 'INT\_OB') >= cast(-32 as INT) AND

CONVERT\_FROM(row\_key, 'INT\_OB') < cast(59 as INT);

有关更多示例，请参阅[测试代码](https://github.com/apache/drill/blob/95623912ebf348962fe8a8846c5f47c5fdcf2f78/contrib/storage-hbase/src/test/java/org/apache/drill/hbase/TestHBaseFilterPushDown.java)。

通过利用有序的字节编码，Drill 1.2和更高版本可以在HBase的big endian数据上执行没有二级索引的条件查询。

## Hive 查询

### 2016年8月4日

这是一个简单的练习，它提供了创建Hive表和插入可以使用Drill查询的数据的步骤。在执行这些步骤之前，请下载[customers.csv](http://doc.mapr.com/download/attachments/28868943/customers.csv?version=1&modificationDate=1426874930765&api=v2)文件。

**注意**

Drill 1.8实现了DROP TABLE和DROP VIEW命令的IF EXISTS参数，使IF成为Drill中的保留字。因此，当您在Hive表的查询中使用它时，必须在Hive的IF条件函数中包含反引号。或者，您可以使用CASE语句而不是IF函数。

要创建Hive表并使用Drill查询，请完成以下步骤：

1. 发出以下命令启动Hive shell：

hive

1. 从Hive shell发出以下命令创建一个表架构：
2. hive> create table customers(FirstName string, LastName string, Company string, Address string, City string, County string, State string, Zip string, Phone string, Fax string, Email string, Web string) row format delimited fields terminated by ',' stored as textfile;
3. 发出以下命令将客户数据加载到customers表中：
4. hive> load data local inpath '/<directory path>/customers.csv' overwrite into table customers;
5. 发出quit或exit离开Hive shell。
6. 启动钻壳。
7. 向Drill发出以下查询以获取Hive表中前十个客户的名字和姓氏：
8. 0: jdbc:drill:schema=hiveremote> SELECT firstname,lastname FROM hiveremote.`customers` limit 10;`

该查询返回以下结果：

+------------+------------+

| firstname | lastname |

+------------+------------+

| Essie | Vaill |

| Cruz | Roudabush |

| Billie | Tinnes |

| Zackary | Mockus |

| Rosemarie | Fifield |

| Bernard | Laboy |

| Sue | Haakinson |

| Valerie | Pou |

| Lashawn | Hasty |

| Marianne | Earman |

+------------+------------+

10 rows selected (1.5 seconds)

0: jdbc:drill:schema=hiveremote>

### 优化实木复合表的读取

使用该store.hive.optimize\_scan\_with\_native\_readers选项优化Hive对Parquet支持的外部表的读取。当设置为TRUE时，此选项使用Drill本地读取器而不是Hive Serde接口，从而导致Parquet支持的外部表的更高性能查询。（Drill 1.2及更高版本）

store.hive.optimize\_scan\_with\_native\_readers按照[“计划和执行选项”](http://drill.apache.org/docs/planning-and-execution-options/)部分所述设置[选项](http://drill.apache.org/docs/planning-and-execution-options/)。

## 复杂查询

### 复杂查询简介

Apache Drill查询不需要事先知道您尝试访问的实际数据，无论其源系统，架构和数据类型如何。Apache Drill的甜蜜点是针对复杂数据的SQL查询工作负载 ：由各种类型的记录和字段组成的数据，而不是以可识别的关系形式（离散的行和列）的数据。在您提交查询时，Drill能够发现数据的形式。嵌套的数据格式，如JSON（JavaScript Object Notation）文件和Parquet文件不仅可以访问：Drill提供了特殊的操作符和函数，您可以使用它们深入到这些文件并提出有趣的分析问题。

这些运营商和功能包括：

* 引用嵌套的数据值
* 访问数组中的数组和数组中的重复值（数组索引）

SQL查询开发人员需要足够好地了解这些数据，以编写查询来识别目标文件中感兴趣的值。例如，编写者需要知道记录是由什么组成的，以及它的数据类型，以便可靠地请求选择列表中正确的“列”。虽然这些数据值不会在源文件中显示为列，但Drill会将它们返回到结果集中，就好像它们在表中具有可预测的列形式一样。Drill还通过将数据视为“柱状”而不是阅读和分析完整记录来优化查询。（Drill使用类似的并行执行和优化功能来处理商业化的MPP数据库。）

鉴于输入文件的基本知识，开发人员需要知道如何使用Drill提供的SQL扩展，以及如何使用它们“嵌入”嵌套的数据。下面的例子展示了如何编写针对JSON文件的简单查询和解压缩嵌套数据的有趣查询。这些示例显示了如何在标准SQL SELECT语句的上下文中使用Drill扩展。在大多数情况下，扩展使用标准JavaScript表示法来引用层次结构中的数据元素。

在你开始之前

本节中的示例使用JSON数据文件。为了编写自己的查询，您需要知道这些文件中的基本数据类型：

* 字符串（双引号内的所有数据），如"0001"or"Cake"
* 数字：整数和浮点数，如0.55或10
* 空值
* 布尔值：true，false

检查您在钻取Web控制台（dfs存储插件配置）中是否具有JSON文件的以下配置设置：

"json" : {

"type" : "json"

}

示例数据：

### 示例数据：甜甜圈

复杂的数据查询使用示例donuts.json文件。要下载此文件，请转到[钻取测试资源](https://github.com/apache/drill/blob/master/exec/java-exec/src/test/resources)页面，在文件列表中找到donuts.json，然后下载它。

这是文件中的第一个“记录”（0001）donuts.json。在查询处理方面，这个记录相当于表中的单个记录。

{

"id": "0001",

"type": "donut",

"name": "Cake",

"ppu": 0.55,

"sales": 35,

"batters":

{

"batter":

[

{ "id": "1001", "type": "Regular" },

{ "id": "1002", "type": "Chocolate" },

{ "id": "1003", "type": "Blueberry" },

{ "id": "1004", "type": "Devil's Food" }

]

},

"topping":

[

{ "id": "5001", "type": "None" },

{ "id": "5002", "type": "Glazed" },

{ "id": "5005", "type": "Sugar" },

{ "id": "5007", "type": "Powdered Sugar" },

{ "id": "5006", "type": "Chocolate with Sprinkles" },

{ "id": "5003", "type": "Chocolate" },

{ "id": "5004", "type": "Maple" }

]

}

数据由地图，数组和嵌套数组组成。名称 - 值对和嵌入的名称/值对定义每个记录的内容。例如， type: donut是一张地图。在下topping，这些对id和type值属于一个数组（在方括号内）。

### 选择平面数据

针对该donuts.json文件的一个非常简单的查询将返回五个“扁平”列的值（仅包含顶级数据的列：不包含嵌套数据）：

0: jdbc:drill:zk=local> select id, type, name, ppu

from dfs.`/Users/brumsby/drill/donuts.json`;

+-------+--------+----------------+-------+

| id | type | name | ppu |

+-------+--------+----------------+-------+

| 0001 | donut | Cake | 0.55 |

| 0002 | donut | Raised | 0.69 |

| 0003 | donut | Old Fashioned | 0.55 |

| 0004 | donut | Filled | 0.69 |

| 0005 | donut | Apple Fritter | 1.0 |

+-------+--------+----------------+-------+

5 rows selected (0.18 seconds)

请注意，dfs是模式名称，文件的路径由反引号括起来，查询必须以分号结尾。

### 使用SQL函数，子句和联接

您可以使用标准的SQL子句（如WHERE和ORDER BY）来详细说明这种简单的查询：

0: jdbc:drill:zk=local> select id, type from dfs.`/Users/brumsby/drill/donuts.json`

where id>0

order by id limit 1;

+------------+------------+

| id | type |

+------------+------------+

| 0001 | donut |

+------------+------------+

1 row selected (0.318 seconds)

您也可以使用标准语法来加入文件（或表格，或文件和表格）：

0: jdbc:drill:zk=local> select tbl1.id, tbl1.type from dfs.`/Users/brumsby/drill/donuts.json` as tbl1

join

dfs.`/Users/brumsby/drill/moredonuts.json` as tbl2

on tbl1.id=tbl2.id;

+------------+------------+

| id | type |

+------------+------------+

| 0001 | donut |

+------------+------------+

1 row selected (0.395 seconds)

也支持等同的USING语法和WHERE子句中的连接。

标准聚合函数针对JSON数据。例如：

0: jdbc:drill:zk=local> select type, avg(ppu) as ppu\_sum from dfs.`/Users/brumsby/drill/donuts.json` group by type;

+------------+------------+

| type | ppu\_sum |

+------------+------------+

| donut | 0.55 |

+------------+------------+

1 row selected (0.216 seconds)

0: jdbc:drill:zk=local> select type, sum(sales) as sum\_by\_type from dfs.`/Users/brumsby/drill/moredonuts.json` group by type;

+------------+-------------+

| type | sum\_by\_type |

+------------+-------------+

| donut | 1194 |

+------------+-------------+

1 row selected (0.389 seconds)

### 为列选择嵌套数据

以下查询显示如何访问记录中不平坦（如topping）的部分内的嵌套数据。要隔离并返回嵌套数据，请使用[n]符号，其中n是指向数组中特定位置的数字。数组使用基于0的索引，因此topping[3]指向数组中的第四个元素topping，而不是第三个元素。

0: jdbc:drill:zk=local> select topping[3] as top from dfs.`/Users/brumsby/drill/donuts.json`;

+------------+

| top |

+------------+

| {"id":"5007","type":"Powdered Sugar"} |

+------------

1 row selected (0.137 seconds)

请注意，此查询为嵌套在文件段内的所有数据生成一列topping。查询写入不解开id和type名称/值对。还要注意为列名使用别名。（如果没有别名，默认的列名就是EXPR$0。）

一些JSON文件将数组存储在数组中。如果您的数据具有此特征，则可以使用以下表示法探测内部数组：[n][n]

例如，假设JSON文件的一部分如下所示：

...

group:

[

[1,2,3],

[4,5,6],

[7,8,9]

]

...

以下查询将返回6（第二个内部数组的第三个值）。

select group[1][2]

### 在嵌套数据中选择多列

以下查询进一步提取JSON数据，从数组内部 选择特定值id和type数据值作为单独的列topping。此查询与前面的查询类似，但是它将值id和type值作为单独的列返回。

0: jdbc:drill:zk=local> select tbl.topping[3].id as record, tbl.topping[3].type as first\_topping

from dfs.`/Users/brumsby/drill/donuts.json` as tbl;

+------------+---------------+

| record | first\_topping |

+------------+---------------+

| 5007 | Powdered Sugar |

+------------+---------------+

1 row selected (0.133 seconds)

该查询还引入了对嵌套数据进行查询的典型要求：使用表别名（本例中名为tbl）。没有表别名，查询将返回一个错误，因为解析器会假定id是一个名为topping的表内的列。和所有标准SQL查询一样，选择tbl.col意味着tbl是现有表的名称（至少在查询期间），col是该表中存在的列。

## 查询系统表

Drill有一个包含系统表的sys数据库。您可以查询系统表以获取有关钻取的信息，包括钻取端口，系统上运行的钻取版本以及可用的钻取选项。在Drill中查看数据库以标识sys数据库，然后使用sys数据库查看可以查询的系统表。

### 查看钻取数据库

发出SHOW DATABASES命令以查看钻取数据库。

0: jdbc:drill:zk=10.10.100.113:5181> show databases;

+--------------------+

| SCHEMA\_NAME |

+--------------------+

| M7 |

| hive.default |

| dfs.default |

| dfs.root |

| dfs.views |

| dfs.tmp |

| dfs.tpcds |

| sys |

| cp.default |

| hbase |

| INFORMATION\_SCHEMA |

+--------------------+

11 rows selected (0.162 seconds)

钻取返回sys数据库结果。

### 使用系统数据库

发出USE命令为后续SQL请求选择sys数据库。

0: jdbc:drill:zk=10.10.100.113:5181> use sys;

+-------+----------------------------------+

| ok | summary |

+-------+----------------------------------+

| true | Default schema changed to [sys] |

+-------+----------------------------------+

1 row selected (0.101 seconds)

### 查看表格

发出SHOW TABLES命令以查看sys数据库中的表。

0: jdbc:drill:zk=10.10.100.113:5181> show tables;

+---------------+-------------+

| TABLE\_SCHEMA | TABLE\_NAME |

+---------------+-------------+

| sys | boot |

| sys | drillbits |

| sys | memory |

| sys | options |

| sys | threads |

| sys | version |

+---------------+-------------+

3 rows selected (0.934 seconds)

0: jdbc:drill:zk=10.10.100.113:5181>

### 查询系统表

查询sys数据库中的drillbits，版本，选项，启动，线程和内存表。

#### 查询钻头表。

0: jdbc:drill:zk=10.10.100.113:5181> select \* from drillbits;

+-------------------+------------+--------------+------------+---------+

| hostname | user\_port | control\_port | data\_port | current|

+-------------------+------------+--------------+------------+--------+

| qa-node115.qa.lab | 31010 | 31011 | 31012 | true |

| qa-node114.qa.lab | 31010 | 31011 | 31012 | false |

| qa-node116.qa.lab | 31010 | 31011 | 31012 | false |

+-------------------+------------+--------------+------------+---------+

3 rows selected (0.146 seconds)

* 主机  
  名称运行Drillbit服务的节点的名称。
* user\_port   
  用户端口地址，用于连接到外部客户端和钻取Web控制台的群集中的节点之间。
* control\_port   
  控制端口地址，用于Apache Drill的多节点安装的节点之间。
* data\_port   
  数据端口地址，用于Apache Drill多节点安装的节点之间。
* 当前  
  True表示Drillbit连接到运行查询的会话或客户端。这个Drillbit是本届会议的工头。

#### 查询版本表。

0: jdbc:drill:zk=10.10.100.113:5181> select \* from version;

+-------------------------------------------+--------------------------------------------------------------------+----------------------------+--------------+----------------------------+

| commit\_id | commit\_message | commit\_time | build\_email | build\_time |

+-------------------------------------------+--------------------------------------------------------------------+----------------------------+--------------+----------------------------+

| d8b19759657698581cc0d01d7038797952888123 | DRILL-3100: TestImpersonationDisabledWithMiniDFS fails on Windows | 15.05.2015 @ 05:18:03 UTC | Unknown | 15.05.2015 @ 06:52:32 UTC |

+-------------------------------------------+--------------------------------------------------------------------+----------------------------+--------------+----------------------------+

1 row selected (0.099 seconds)

* commit\_id   
  你正在运行的版本的github ID。例如，
* commit\_message   
  解释更改的消息。
* commit\_time更改  
  的日期和时间。
* build\_email   
  进行更改的人员的电子邮件地址，在本例中未知。
* build\_time   
  发布版本的时间。

#### 查询选项表。

Drill提供您可以查询的系统，会话和启动选项。

以下示例显示了有关系统选项的查询：

0: jdbc:drill:zk=10.10.100.113:5181> select \* from options where type='SYSTEM' limit 10;

+-------------------------------------------------+----------+---------+----------+-------------+-------------+-----------+------------+

| name | kind | type | status | num\_val | string\_val | bool\_val | float\_val |

+-------------------------------------------------+----------+---------+----------+-------------+-------------+-----------+------------+

| drill.exec.functions.cast\_empty\_string\_to\_null | BOOLEAN | SYSTEM | DEFAULT | null | null | false | null |

| drill.exec.storage.file.partition.column.label | STRING | SYSTEM | DEFAULT | null | dir | null | null |

| exec.errors.verbose | BOOLEAN | SYSTEM | DEFAULT | null | null | false | null |

| exec.java\_compiler | STRING | SYSTEM | DEFAULT | null | DEFAULT | null | null |

| exec.java\_compiler\_debug | BOOLEAN | SYSTEM | DEFAULT | null | null | true | null |

| exec.java\_compiler\_janino\_maxsize | LONG | SYSTEM | DEFAULT | 262144 | null | null | null |

| exec.max\_hash\_table\_size | LONG | SYSTEM | DEFAULT | 1073741824 | null | null | null |

| exec.min\_hash\_table\_size | LONG | SYSTEM | DEFAULT | 65536 | null | null | null |

| exec.queue.enable | BOOLEAN | SYSTEM | DEFAULT | null | null | false | null |

| exec.queue.large | LONG | SYSTEM | DEFAULT | 10 | null | null | null |

+-------------------------------------------------+----------+---------+----------+-------------+-------------+-----------+------------+

10 rows selected (0.216 seconds)

* 名称  
  选项的名称。
* kind   
  选项值的数据类型。
* 键入  
  输出中的选项类型：系统或会话。
* 状态选项的状态：默认或更改。
* num\_val   
  默认值，它是long或int数据类型; 否则为空。
* string\_val   
  默认值，它是一个字符串; 否则为空。
* bool\_val   
  默认值，即true或false; 否则为空。
* float\_val   
  默认值，它是double，float或long double数据类型的默认值; 否则为空。

#### 查询启动表。

0: jdbc:drill:zk=10.10.100.113:5181> select \* from boot limit 10;

+--------------------------------------+----------+-------+---------+------------+-------------------------+-----------+------------+

| name | kind | type | status | num\_val | string\_val | bool\_val | float\_val |

+--------------------------------------+----------+-------+---------+------------+-------------------------+-----------+------------+

| awt.toolkit | STRING | BOOT | BOOT | null | "sun.awt.X11.XToolkit" | null | null |

| drill.client.supports-complex-types | BOOLEAN | BOOT | BOOT | null | null | true | null |

| drill.exec.buffer.size | STRING | BOOT | BOOT | null | "6" | null | null |

| drill.exec.buffer.spooling.delete | BOOLEAN | BOOT | BOOT | null | null | true | null |

| drill.exec.buffer.spooling.size | LONG | BOOT | BOOT | 100000000 | null | null | null |

| drill.exec.cluster-id | STRING | BOOT | BOOT | null | "SKCluster" | null | null |

| drill.exec.compile.cache\_max\_size | LONG | BOOT | BOOT | 1000 | null | null | null |

| drill.exec.compile.compiler | STRING | BOOT | BOOT | null | "DEFAULT" | null | null |

| drill.exec.compile.debug | BOOLEAN | BOOT | BOOT | null | null | true | null |

| drill.exec.compile.janino\_maxsize | LONG | BOOT | BOOT | 262144 | null | null | null |

+--------------------------------------+----------+-------+---------+------------+-------------------------+-----------+------------+

10 rows selected (0.192 seconds)

* 名称  
  启动选项的名称。
* kind   
  选项值的数据类型。
* 键入  
  这总是启动。
* 状态这总是启动。
* num\_val   
  默认值，它是long或int数据类型; 否则为空。
* string\_val   
  默认值，它是一个字符串; 否则为空。
* bool\_val   
  默认值，即true或false; 否则为空。
* float\_val   
  默认值，它是double，float或long double数据类型的默认值; 否则为空。

#### 查询线程表。

0: jdbc:drill:zk=10.10.100.113:5181> select \* from threads;

+--------------------+------------+----------------+---------------+

| hostname | user\_port | total\_threads | busy\_threads |

+--------------------+------------+----------------+---------------+

| qa-node115.qa.lab | 31010 | 33 | 33 |

| qa-node114.qa.lab | 31010 | 33 | 32 |

| qa-node116.qa.lab | 31010 | 29 | 29 |

+--------------------+------------+----------------+---------------+

3 rows selected (0.618 seconds)

* 主机  
  名称运行Drillbit服务的节点的名称。
* user\_port   
  用户端口地址，用于连接到外部客户端和钻取Web控制台的群集中的节点之间。
* total\_threads节点上的最高线程数。
* busy\_threads节点上的活动线程（守护进程和非守护进程）的当前数量。

#### 查询内存表。

0: jdbc:drill:zk=10.10.100.113:5181> select \* from memory;

+--------------------+------------+---------------+-------------+-----------------+---------------------+-------------+

| hostname | user\_port | heap\_current | heap\_max | direct\_current | jvm\_direct\_current | direct\_max |

+--------------------+------------+---------------+-------------+-----------------+---------------------+-------------+

| qa-node115.qa.lab | 31010 | 443549712 | 4294967296 | 11798941 | 167772974 | 8589934592 |

| qa-node114.qa.lab | 31010 | 149948432 | 4294967296 | 7750365 | 134218542 | 8589934592 |

| qa-node116.qa.lab | 31010 | 358612992 | 4294967296 | 7750365 | 83886894 | 8589934592 |

+--------------------+------------+---------------+-------------+-----------------+---------------------+-------------+

3 rows selected (0.172 seconds)

* 主机  
  名称运行Drillbit服务的节点的名称。
* user\_port   
  用户端口地址，用于连接到外部客户端和钻取Web控制台的群集中的节点之间。
* heap\_current在堆上使用的内存量，以字节为单位。
* heap\_max堆中可用的最大内存量，以字节为单位。
* direct\_current分配器使用的当前直接内存，以字节为单位。
* jvm\_direct\_current当前的JVM直接内存分配，以字节为单位。
* direct\_max分配器可用的最大直接内存，以字节为单位。

有关如何配置钻取系统和会话选项的信息，请参阅[计划和执行选项](http://drill.apache.org/docs/planning-and-execution-options)。

有关如何配置Drill启动选项的信息，请参阅[启动选项](http://drill.apache.org/docs/start-up-options)。

性能调优

# 性能调整

* [性能调整介绍](http://drill.apache.org/docs/performance-tuning-introduction/)
* [分区修剪](http://drill.apache.org/docs/partition-pruning/)
  + [分区修剪简介](http://drill.apache.org/docs/partition-pruning-introduction/)
  + [如何分区数据](http://drill.apache.org/docs/how-to-partition-data/)
* [异步拼花阅读器](http://drill.apache.org/docs/asynchronous-parquet-reader/)
* [优化拼花元数据阅读](http://drill.apache.org/docs/optimizing-parquet-metadata-reading/)
* [实木复合地板过滤器下推](http://drill.apache.org/docs/parquet-filter-pushdown/)
* [Hive元数据缓存](http://drill.apache.org/docs/hive-metadata-caching/)
* [选择存储格式](http://drill.apache.org/docs/choosing-a-storage-format/)
* [查询计划和调整](http://drill.apache.org/docs/query-plans-and-tuning/)
  + [查询计划和调整介绍](http://drill.apache.org/docs/query-plans-and-tuning-introduction/)
  + [加入计划指南](http://drill.apache.org/docs/join-planning-guidelines/)
  + [优化聚合指南](http://drill.apache.org/docs/guidelines-for-optimizing-aggregation/)
  + [修改查询计划选项](http://drill.apache.org/docs/modifying-query-planning-options/)
  + [基于排序和哈希的内存约束运算符](http://drill.apache.org/docs/sort-based-and-hash-based-memory-constrained-operators/)
  + [启用查询队列](http://drill.apache.org/docs/enabling-query-queuing/)
  + [节流](http://drill.apache.org/docs/throttling/)
  + [控制并行化以平衡性能与多租户](http://drill.apache.org/docs/controlling-parallelization-to-balance-performance-with-multi-tenancy/)
* [识别性能问题](http://drill.apache.org/docs/identifying-performance-issues/)
  + [查询计划](http://drill.apache.org/docs/query-plans/)
  + [查询配置文件](http://drill.apache.org/docs/query-profiles/)
* [性能调优参考](http://drill.apache.org/docs/performance-tuning-reference/)
  + [查询配置文件列说明](http://drill.apache.org/docs/query-profile-column-descriptions/)
  + [物理运营商](http://drill.apache.org/docs/physical-operators/)
* [监测指标](http://drill.apache.org/docs/monitoring-metrics/)

# 数据源和文件格式

* [数据源和文件格式介绍](http://drill.apache.org/docs/data-sources-and-file-formats-introduction/)
* [Hive-Drill数据类型映射](http://drill.apache.org/docs/hive-to-drill-data-type-mapping/)
* [部署和使用Hive UDF](http://drill.apache.org/docs/deploying-and-using-a-hive-udf/)
* [镶木地板格式](http://drill.apache.org/docs/parquet-format/)
* [JSON数据模型](http://drill.apache.org/docs/json-data-model/)
* [文本文件：CSV，TSV，PSV](http://drill.apache.org/docs/text-files-csv-tsv-psv/)
* [序列文件](http://drill.apache.org/docs/sequence-files/)

## 数据源和文件格式介绍：

### 2016年5月5日

包含在Drill支持的数据源中的是这些关键数据源：

* HBase的
* 蜂巢
* MAPR-DB
* 文件系统

Drill认为数据源具有强大的模式或弱的模式。

下表介绍了每种模式类型：

| **架构类型** | **描述** |
| --- | --- |
| 强大 | 除文本文件数据源外，Drill验证与强模式关联的数据源包含与查询中使用的数据类型兼容的数据类型。Drill还会验证查询中引用的列是否存在于基础数据源中。如果列不存在，CREATE VIEW将失败。 |
| 弱 | Drill不会验证与弱架构相关联的数据源是否包含与查询中使用的数据类型兼容的数据类型。Drill不验证在Parquet数据源的查询中引用的列是否存在，因此CREATE VIEW总是成功。在JSON文件的情况下，Drill不验证文件是否包含视图中指定的映射。 |

下表列出了模式的当前类别以及与其相关的数据源：

|  | **强大的架构** | **模式弱** |
| --- | --- | --- |
| 数据源 | 视图，配置单元表，hbase列族，文本 | json，mongodb，hbase列限定符，实木复合地板 |

### Drill支持以下数据输入格式：

* [Avro公司](http://avro.apache.org/docs/current/spec.html)
* CSV（逗号分隔值）
* TSV（制表符分隔值）
* PSV（管道分离值）
* 实木复合地板
* MAPR-DB \*
* Hadoop序列文件

\*仅在使用mapr-drill软件包在群集上安装Drill时才可用。

您将[存储插件](http://drill.apache.org/docs/storage-plugin-registration)定义的工作空间部分中来自数据源的数据的输入格式设置为Drill 。Drill中默认的输入格式是Parquet。

您更改[sys.options表](http://drill.apache.org/docs/configuration-options-introduction/)store中的一个属性以设置Drill数据的输出格式。Drill CREATE TABLE AS（CTAS）语句的默认存储格式是Parquet。

## Parquet 数据格式：

2017年8月9日

[Apache Parquet](http://parquet.incubator.apache.org/documentation/latest)具有以下特点：

* 自描述
* 列格式
* 与语言无关

自描述数据将数据本身嵌入到模式或结构中。Hadoop用例推动自描述数据格式（如Parquet和JSON）以及NoSQL数据库（如HBase）的增长。这些格式和数据库非常适合Hadoop应用程序和BI /分析的敏捷和迭代开发周期。为了处理大文件而进行了优化，Parquet将列中的数据进行排列，将相关值彼此靠近以优化查询性能，最大限度地减少I / O并促进压缩。实木复合地板使用节省资源的技术检测和编码相同或相似的数据。

Apache Drill包含以下对Parquet的支持：

* 查询文件或NoSQL数据库中的自描述数据，而无需在集中式转换中定义和管理模式覆盖定义
* 从其他文件格式（如JSON）创建Parquet文件，无需任何设置
* 生成具有演变或改变模式的Parquet文件，并实时查询数据
* 处理Parquet数据类型

### Parquet File

当读取Parquet数据时，Drill只加载必要的数据列，从而减少I / O。Drill可以从数据文件或表中只读取一小部分Parquet数据，可以检查和分析跨多个文件的列的所有值。您可以创建一个格式的钻表，并以另一种格式存储数据，包括Parquet。

### 写parquet文件

CREATE TABLE AS（CTAS）可以使用存储插件提供的任何数据源。要使用CTAS命令写入Parquet数据，请设置session store.format选项，如[配置Parquet存储格式中所示](http://drill.apache.org/docs/parquet-format/#configuring-the-parquet-storage-format)。或者，将存储插件配置为指向包含Parquet文件的目录。

尽管数据驻留在单个表中，但Parquet输出通常由多个文件组成，这些文件类似于具有编号文件名的MapReduce输出，如目录中的0\_0\_0.parquet。

### 日期值自动更正

从Drill 1.10开始，Drill写入标准Parquet日期值。Drill还具有自动修正功能，可自动检测并修正Drill 1.10之前Drill写入Parquet文件的损坏日期值。

默认情况下，自动更正功能已打开，并可在未来5000年的时间内工作。在极少的情况下，Drill需要将日期写入将来数千年，关闭自动更正功能。

要禁用自动更正功能，请导航到存储插件配置，autoCorrectCorruptDates并将Parquet配置中的选项更改为“false”，如下例所示：

"formats": {

"parquet": {

"type": "parquet",

"autoCorrectCorruptDates": false

}

或者，您可以在发出查询时将该选项设置为false，如以下示例所示：

SELECT l\_shipdate, l\_commitdate FROM table(dfs.`/drill/testdata/parquet\_date/dates\_nodrillversion/drillgen2\_lineitem`

(type => 'parquet', autoCorrectCorruptDates => false)) LIMIT 1;

### 配置Parquet存储格式

要读取或写入Parquet数据，您需要在存储插件格式定义中包含Parquet格式。该dfs插件定义包括镶木格式。

使用该store.format选项在会话或系统级别设置Parquet行组的CTAS输出格式。

使用ALTER命令设置store.format选项。

ALTER SYSTEM|SESSION SET `store.format` = 'parquet';

### 配置Parquet文件的大小

通过设置配置Parquet文件的大小store.parquet.block-size可以提高写入性能。块大小是MFS，HDFS或文件系统的大小。

Drill需要缓冲数据的块越大。包含单个块的镶木地板文件使得数据量最大化，连续地存储在磁盘上。给定每个文件的单个行组，Drill将整个Parquet文件存储到块中，从而避免网络I / O。

要使性能最大化，请使用该store.parquet.block-size选项将Parquet行组的目标大小设置为小于或等于MFS，HDFS块大小或文件系统的字节数，如下所示：

ALTER SYSTEM|SESSION SET `store.parquet.block-size` = 536870912;

默认的块大小是536870912字节。

### 配置Parquet文件的HDFS块大小

Drill 1.11引入了这个store.parquet.writer.use\_single\_fs\_block选项，它使Drill能够在不更改默认文件系统块大小的情况下将Parquet文件写入单个文件系统块。当Drill将Parquet文件作为文件系统上的一个块读取时，查询性能会提高。当store.parquet.writer.use\_single\_fs\_block启用该选项时，该store.parquet.block-size设置确定创建的Parquet文件的块大小。该store.parquet.writer.use\_single\_fs\_block选项的默认设置是“false”。使用SET命令启用或禁用该选项，如下所示：

ALTER SYSTEM|SESSION SET store.parquet.writer.use\_single\_fs\_block = 'true|false';

### 类型映射

Parquet和SQL数据类型之间的高度相关性使得在Drill中轻松阅读Parquet文件。写入Parquet文件需要比阅读更多的工作。由于SQL不支持所有的Parquet数据类型，为了防止Drill推断除你想要的类型以外的类型，使用转换[函数](http://drill.apache.org/docs/data-type-conversion/#cast)如果Parquet数据是逻辑类型，则Drill为Parquet转换提供比SQL更多的自由转换功能。

以下一般过程将文件从JSON转换为Parquet：

* 创建或使用现有的存储插件，指定Parquet文件的存储位置，数据的可变性和支持的文件格式。
* 看看JSON数据。
* 创建一个选择JSON文件的表。
* 在CTAS命令中，将JSON字符串数据转换为相应的[SQL类型](http://drill.apache.org/docs/json-data-model/#data-type-mapping)。

### 例如：读取JSON，写入实木复合地板

此示例演示存储插件定义，来自JSON文件的示例数据行以及将JSON输入写入Parquet输出的Drill查询。

### 存储插件定义

您可以使用与Drill一起安装的默认dfs存储插件来读写Parquet文件。存储插件需要将工作区的可写选项配置为true，所以Drill可以写入Parquet输出。dfs存储插件定义了tmp可写入的工作区，您可以在CTAS命令中使用该工作区来创建Parquet表。

### JSON数据示例行

名为sample.json的JSON文件包含由JSON数据的典型字符串组成的数据。以下示例显示了JSON文件的一行：

{"trans\_id":0,"date":"2013-07-26","time":"04:56:59","amount":80.5,"user\_info":

{"cust\_id":28,"device":"WEARABLE2","state":"mt"

},"marketing\_info":

{"camp\_id":4,"keywords": ["go","to","thing","watch","made","laughing","might","pay","in","your","hold"]

},

"trans\_info":

{"prod\_id":[16],

"purch\_flag":"false"

}

}

### CTAS查询

以下示例显示一个CTAS查询，该查询根据上一个示例中显示的JSON数据创建一个表。该命令将日期，时间和数量字符串转换为SQL类型DATE，TIME和DOUBLE。其他字符串的字符串到VARCHAR转换自动发生。

CREATE TABLE dfs.tmp.sampleparquet AS

(SELECT trans\_id,

cast(`date` AS date) transdate,

cast(`time` AS time) transtime,

cast(amount AS double) amountm,

user\_info, marketing\_info, trans\_info

FROM dfs.`/Users/drilluser/sample.json`);

CTAS查询不指定输出的文件扩展名。Drill默认创建一个parquet文件，如输出中的文件名所示：

+------------+---------------------------+

| Fragment | Number of records written |

+------------+---------------------------+

| 0\_0 | 5 |

+------------+---------------------------+

1 row selected (1.369 seconds)

您可以查询Parquet文件以验证Drill现在将转换的字符串解释为日期。

SELECT extract(year from transdate) AS `Year`, t.user\_info.cust\_id AS Customer

FROM dfs.tmp.`sampleparquet` t;

+------------+------------+

| Year | Customer |

+------------+------------+

| 2013 | 28 |

| 2013 | 86623 |

| 2013 | 11 |

| 2013 | 666 |

| 2013 | 999 |

+------------+------------+

5 rows selected (0.039 seconds)

有关使用Parquet数据的更多示例和信息，请参阅[“](https://www.mapr.com/blog/evolving-parquet-self-describing-data-format-new-paradigms-consumerization-hadoop-data#.VNeqQbDF_8f)将Parquet [演化为自描述数据格式 - Hadoop数据消费化的新范例”](https://www.mapr.com/blog/evolving-parquet-self-describing-data-format-new-paradigms-consumerization-hadoop-data#.VNeqQbDF_8f)。

### SQL数据类型实木复合地板

本节中的第一个表将SQL数据类型映射到Parquet数据类型，由Parquet创建者有意限制，以最大限度地减少对磁盘存储的影响：

| **SQL Type** | **Parquet Type** | **Description** |
| --- | --- | --- |
| BIGINT | INT64 | 8-byte signed integer |
| BOOLEAN | BOOLEAN | TRUE (1) or FALSE (0) |
| N/A | BYTE\_ARRAY | Arbitrarily long byte array |
| FLOAT | FLOAT | 4-byte single precision floating point number |
| DOUBLE | DOUBLE | 8-byte double precision floating point number |
| INTEGER | INT32 | 4-byte signed integer |
| VARBINARY(12)\* | INT96 | 12-byte signed int |

\* Drill 1.10及更高版本可以隐式地将Parquet INT96类型解释为TIMESTAMP（具有标准的8字节/毫秒精度），当store.parquet.reader.int96\_as\_timestamp启用该选项时。在Drill（1.2到1.9）的早期版本中或者store.parquet.reader.int96\_as\_timestamp禁用选项时，您必须使用Drill的CONVERT\_FROM函数将INT96值正确解释为TIMESTAMP值。

### 关于INT96支持

从Drill 1.10开始，当store.parquet.reader.int96\_as\_timestamp启用该选项时，Drill可以隐式地解释Parquet文件中的INT96时间戳数据类型。对于Drill的早期版本，或者store.parquet.reader.int96\_as\_timestamp禁用选项时，您必须使用CONVERT\_FROM函数，

该store.parquet.reader.int96\_as\_timestamp选项默认是禁用的。使用[ALTER SYSTEM | SESSION SET](http://drill.apache.org/docs/alter-system/)命令启用该选项。不必要地启用此选项可能会导致查询失败，因为CONVERT\_FROM（col，'TIMESTAMP\_IMPALA'）功能在store.parquet.reader.int96\_as\_timestamp启用时不起作用。

### 使用CONVERT\_FROM来解释INT96

在较早版本的Drill（1.2到1.9）中，必须使用CONVERT\_FROM函数进行Drill来解释Parquet INT96类型。例如，要解码来自Hive或Impala（INT96类型）的时间戳，请使用CONVERT\_FROM函数和[TIMESTAMP\_IMPALA](http://drill.apache.org/docs/supported-data-types/#data-types-for-convert_to-and-convert_from-functions)类型参数：

SELECT CONVERT\_FROM(timestamp\_field, 'TIMESTAMP\_IMPALA') as timestamp\_field FROM `dfs.file\_with\_timestamp.parquet`;

由于INT96仅支持只读，所以不能将TIMESTAMP\_IMPALA用作CONVERT\_TO的数据类型参数。您可以使用CAST函数将SQL TIMESTAMP转换为VARBINARY，但生成的VARBINARY与INT96不同。

例如，在读取INT96并将一些数据转换为时间戳后创建一个钻取表。

CREATE TABLE t2(c1) AS SELECT CONVERT\_FROM(created\_ts, 'TIMESTAMP\_IMPALA') FROM t1 ORDER BY 1 LIMIT 1;

t1.created\_ts是INT96（或Hive / Impala时间戳），t2.created\_ts是SQL时间戳。这些类型是不可比较的。你不能使用像t1.created\_ts = t2.created\_ts这样的条件。

### 配置时区

默认情况下，INT96时间戳值表示本地日期和时间，与Hive类似。要获得UTC的INT96时间戳值，请配置钻取[UTC时间](http://drill.apache.org/docs/data-type-conversion/#time-zone-limitation)。

### SQL类型实例逻辑类型

Parquet还支持逻辑类型，在[Apache Parquet站点](https://github.com/Parquet/parquet-format/blob/master/LogicalTypes.md)上进行了充分描述。嵌入式类型，JSON和BSON，注释表示JSON或BSON文档的二进制基本类型。逻辑类型及其对SQL类型的映射如下

| **SQL Type** | **Drill Description** | **Parquet Logical Type** | **Parquet Description** |
| --- | --- | --- | --- |
| DATE | Years months and days in the form in the form YYYY-­MM-­DD | DATE | Date, not including time of day. Uses the int32 annotation. Stores the number of days from the Unix epoch, 1 January 1970. |
| VARCHAR | Character string variable length | UTF8 (Strings) | Annotates the binary primitive type. The byte array is interpreted as a UTF-8 encoded character string. |
| None |  | INT\_8 | 8 bits, signed |
| None |  | INT\_16 | 16 bits, usigned |
| INT | 4-byte signed integer | INT\_32 | 32 bits, signed |
| DOUBLE | 8-byte double precision floating point number | INT\_64 | 64 bits, signed |
| None |  | UINT\_8 | 8 bits, unsigned |
| None |  | UINT\_16 | 16 bits, unsigned |
| None |  | UINT\_32 | 32 bits, unsigned |
| None |  | UINT\_64 | 64 bits, unsigned |
| DECIMAL\* | 38-digit precision | DECIMAL | Arbitrary-precision signed decimal numbers of the form unscaledValue \* 10^(-scale) |
| TIME | Hours, minutes, seconds, milliseconds; 24-hour basis | TIME\_MILLIS | Logical time, not including the date. Annotates int32. Number of milliseconds after midnight. |
| TIMESTAMP | Year, month, day, and seconds | TIMESTAMP\_MILLIS | Logical date and time. Annotates an int64 that stores the number of milliseconds from the Unix epoch, 00:00:00.000 on 1 January 1970, UTC. |
| INTERVAL | Integer fields representing a period of time depending on the type of interval | INTERVAL | An interval of time. Annotates a fixed\_len\_byte\_array of length 12. Months, days, and ms in unsigned little-endian encoding. |

\*在此版本中，Drill将禁用DECIMAL数据类型，包括转换为DECIMAL并从Parquet和Hive读取DECIMAL类型。要启用DECIMAL类型，请将该planner.enable\_decimal\_data\_type选项设置为true。

### 数据描述语言支持

Parquet支持以下数据描述语言：

* 阿帕奇Avro
* Apache Thrift
* Google协议缓冲区

实施自定义存储插件，为Thrift等格式创建Parquet读写器。

# ODBC / JDBC接口

* [接口介绍](http://drill.apache.org/docs/interfaces-introduction/)
* [使用JDBC驱动程序](http://drill.apache.org/docs/using-the-jdbc-driver/)
* [在Windows上使用JDBC和SQuirreL](http://drill.apache.org/docs/using-jdbc-with-squirrel-on-windows/)
* [安装ODBC驱动程序](http://drill.apache.org/docs/installing-the-odbc-driver/)
  + [在Linux上安装驱动程序](http://drill.apache.org/docs/installing-the-driver-on-linux/)
  + [在Mac OS X上安装驱动程序](http://drill.apache.org/docs/installing-the-driver-on-mac-os-x/)
  + [在Windows上安装驱动程序](http://drill.apache.org/docs/installing-the-driver-on-windows/)
* [配置ODBC](http://drill.apache.org/docs/configuring-odbc/)
  + [ODBC配置参考](http://drill.apache.org/docs/odbc-configuration-reference/)
  + [记录和追踪](http://drill.apache.org/docs/logging-and-tracing/)
  + [在Linux上配置ODBC](http://drill.apache.org/docs/configuring-odbc-on-linux/)
  + [在Mac OS X上配置ODBC](http://drill.apache.org/docs/configuring-odbc-on-mac-os-x/)
  + [在Windows上配置ODBC](http://drill.apache.org/docs/configuring-odbc-on-windows/)
  + [测试ODBC连接](http://drill.apache.org/docs/testing-the-odbc-connection/)
* [使用钻探资源管理器](http://drill.apache.org/docs/using-drill-explorer/)
  + [钻探浏览器介绍](http://drill.apache.org/docs/drill-explorer-introduction/)
  + [将浏览器连接到数据](http://drill.apache.org/docs/connecting-drill-explorer-to-data/)
  + [浏览数据和定义视图](http://drill.apache.org/docs/browsing-data-and-defining-views/)
* [使用BI工具进行钻取](http://drill.apache.org/docs/using-drill-with-bi-tools/)
  + [使用BI工具进行钻取简介](http://drill.apache.org/docs/using-drill-with-bi-tools-introduction/)
  + [Tableau示例](http://drill.apache.org/docs/tableau-examples/)
  + [通过Apache Drill使用MicroStrategy Analytics](http://drill.apache.org/docs/using-microstrategy-analytics-with-apache-drill/)
  + [使用Tibco Spotfire桌面与钻取](http://drill.apache.org/docs/using-tibco-spotfire-desktop-with-drill/)
  + [使用Drill配置Tibco Spotfire服务器](http://drill.apache.org/docs/configuring-tibco-spotfire-server-with-drill/)
  + [使用Qlik Sense和Drill](http://drill.apache.org/docs/using-qlik-sense-with-drill/)
  + [使用Apache Drill与Tableau 10.2](http://drill.apache.org/docs/using-apache-drill-with-tableau-10-2/)
  + [将Apache Drill与Tableau 9 Desktop一起使用](http://drill.apache.org/docs/using-apache-drill-with-tableau-9-desktop/)
  + [使用Apache Drill与Tableau 9服务器](http://drill.apache.org/docs/using-apache-drill-with-tableau-9-server/)
  + [通过Apache Drill使用Information Builders的WebFOCUS](http://drill.apache.org/docs/using-information-builders-webfocus-with-apache-drill/)
  + [使用钻取来配置JReport](http://drill.apache.org/docs/configuring-jreport-with-drill/)

## 接口介绍

### 2017年8月18日

您可以通过以下接口连接到Apache Drill：

* 钻壳
* 钻取Web控制台
* [ODBC](http://drill.apache.org/docs/installing-the-odbc-driver/) \*
* [JDBC](http://drill.apache.org/docs/using-jdbc-with-squirrel-on-windows/)
* C ++ API

\* Apache Drill没有开源的ODBC驱动。但是，MapR提供了专门为将Apache Drill连接到BI工具而开发的ODBC驱动程序。

### 使用ODBC从BI工具访问Apache Drill

MapR提供了一个将Windows，Mac OS X和Linux连接到Apache Drill和BI工具的ODBC驱动程序。使用最新版本的MapR Drill ODBC驱动程序安装最新版本的Apache Drill。

在[http://package.mapr.com/tools/MapR-ODBC/MapR\_Drill/上](http://package.mapr.com/tools/MapR-ODBC/MapR_Drill/)访问最新的MapR Drill ODBC驱动程序。

### 使用JDBC从SQuirrel访问Apache Drill

您可以在Windows，Linux和Mac OS X系统上通过JDBC客户端工具（如SQuirreL）连接到Drill，以访问在Drill中注册的所有数据源。Drill中包含嵌入式JDBC驱动程序。在SQuirreL客户端中配置JDBC驱动程序以连接到SQuirreL的Drill。本节提供从Windows上的SQuirreL连接到Drill的说明。

要在Windows上使用带有SQuirreL的Drill JDBC驱动程序，请完成以下步骤：

* [第1步：获取钻取JDBC驱动程序](http://drill.apache.org/docs/using-the-jdbc-driver/#getting-the-drill-jdbc-driver)
* [第2步：安装和启动SQuirreL](http://drill.apache.org/docs/using-jdbc-with-squirrel-on-windows/#step-2:-installing-and-starting-squirrel)
* [第3步：将钻取JDBC驱动程序添加到SQuirreL](http://drill.apache.org/docs/using-jdbc-with-squirrel-on-windows/#step-3:-adding-the-drill-jdbc-driver-to-squirrel)
* [第4步：从SQuirreL运行查询](http://drill.apache.org/docs/using-jdbc-with-squirrel-on-windows/#step-4:-running-a-drill-query-from-squirrel)

有关如何使用SQuirreL的信息，请参阅[SQuirreL快速入门](http://squirrel-sql.sourceforge.net/user-manual/quick_start.html) 指南。

## 使用JDBC驱动程序

2017年5月9日

本节介绍如何安装和使用Apache Drill的JDBC驱动程序。要使用JDBC驱动程序，您必须：

1. [满足先决条件](http://drill.apache.org/docs/using-the-jdbc-driver/#prerequisites)。
2. [获取钻取JDBC驱动程序](http://drill.apache.org/docs/using-the-jdbc-driver/#getting-the-drill-jdbc-driver)。
3. 将Drill JDBC jar文件放在类路径中。
4. 编写应用程序代码或配置BI工具时，请在JDBC连接字符串中使用[有效的URL](http://drill.apache.org/docs/using-the-jdbc-driver/#using-the-jdbc-url-for-a-random-drillbit-connection)。
5. 在应用程序代码或客户端工具的配置中使用[Drill驱动程序类](http://drill.apache.org/docs/using-the-jdbc-driver/#using-the-drill-driver-class-name)的[名称](http://drill.apache.org/docs/using-the-jdbc-driver/#using-the-drill-driver-class-name)。

大多数客户端工具提供了一个UI，您可以在其中输入所有必需的连接信息，包括驱动程序位置，连接URL和驱动程序类名称。有关通过JDBC进行钻取的客户端工具连接的特定示例，请参阅[在SQuirreL中使用JDBC](http://drill.apache.org/docs/using-jdbc-with-squirrel-on-windows)和[配置Spotfire Server](http://drill.apache.org/docs/configuring-tibco-spotfire-server-with-drill/)。

### 先决条件

* JRE 7或JDK 7
* [一个钻子安装](http://drill.apache.org/docs/install-drill/)
* 能够使用IP解析Drill节点的实际主机名，如在[使用JReport安装Drill JDBC驱动程序的](http://drill.apache.org/docs/configuring-jreport-with-drill/#step-1:-install-the-drill-jdbc-driver-with-jreport)步骤4中[所述](http://drill.apache.org/docs/configuring-jreport-with-drill/#step-1:-install-the-drill-jdbc-driver-with-jreport)。

### 获取钻取JDBC驱动程序

钻取JDBC驱动程序JAR文件必须存在于客户端计算机上，以便您可以配置打算使用的应用程序或第三方工具的驱动程序。通过以下方式之一获取驱动程序：

* 将drill-jdbc-allJAR文件从安装Drill的节点上的以下Drill安装目录复制到客户端计算机上的目录中：  
  <drill\_installation\_directory>/jars/jdbc-driver/drill-jdbc-all-<version>.jar

要么

* [将最新的Drill版本](http://apache.osuosl.org/drill/)的[TAR文件](http://apache.osuosl.org/drill/)下载到客户机上的某个位置，然后解压缩该文件。在Windows上，您可能需要使用解压缩实用程序，例如[7-zip](http://www.7-zip.org/)。驱动程序被提取到以下目录：  
  <drill-installation\_directory>/jars/jdbc-driver/drill-jdbc-all-<version>.jar

### 使用随机钻取连接的JDBC URL

JDBC URL的格式略有不同，取决于您想要连接到Drillbit的方式：随机，本地还是直接。本节介绍如何使用URL进行随机或本地连接。[稍后](http://drill.apache.org/docs/using-the-jdbc-driver/#using-the-jdbc-url-format-for-a-direct-drillbit-connection)会介绍使用URL [直接连接到Drillbit](http://drill.apache.org/docs/using-the-jdbc-driver/#using-the-jdbc-url-format-for-a-direct-drillbit-connection)。如果您希望ZooKeeper在群集中随机选择一个Drillbit，或者要连接到本地Drillbit，则驱动程序URL的格式为：

jdbc:drill:zk=<zk name>[:<port>][,<zk name2>[:<port>]...  
<directory>/<cluster ID>;[schema=<storage plugin>]

哪里

* jdbc是连接类型。需要。
* schema是用作查询默认值的[存储插件](http://drill.apache.org/docs/storage-plugin-registration)配置的名称。例如schema=hive。可选的。
* zk name指定一个或多个ZooKeeper主机名或IP地址。使用local主机名或IP地址代替本地的Drillbit。需要。
* port是ZooKeeper端口号。端口2181是默认值。在MapR群集上，缺省值为5181.可选。
* directory是ZooKeeper中的Drill目录，默认为/Drill。可选的。
* cluster ID是drillbits1默认的。如果默认值已更改，请[确定群集ID](http://drill.apache.org/docs/using-the-jdbc-driver/#determining-the-cluster-id)并使用它。可选的。

### 确定群集ID

要确定群集ID，请检查以下文件：

<drill-installation>/conf/drill-override.conf

例如：

...

drill.exec: {

cluster-id: "docs41cluster-drillbits",

zk.connect: "centos23.lab:5181,centos28.lab:5181,centos29.lab:5181"

}

...

### URL示例

**单节点安装**

jdbc:drill:zk=maprdemo:5181

jdbc:drill:zk=centos23.lab:2181/drill/docs41cluster-drillbits

jdbc:drill:zk=10.10.100.56:2181/drill/drillbits1;schema=hive

**群集安装**

jdbc:drill:zk=10.10.100.30:5181,10.10.100.31:5181,10.10.100.32:2181/drill/drillbits1;schema=hive

### 使用JDBC URL格式进行直接Drillbit连接

如果您想直接连接到一个钻头，而不是使用ZooKeeper的选择钻头，替代zk=<zk name>与drillbit=<node name>下面的网址如下所示：

jdbc:drill:drillbit=<node name>[:<port>][,<node name2>[:<port>]...  
<directory>/<cluster ID>[schema=<storage plugin>]

哪里

drillbit=<node name> 指定运行Drill的群集节点的一个或多个主机名或IP地址。

### tries 参数

从Drill 1.10开始，可以tries=<value>在连接字符串中包含可选参数，如以下URL所示：

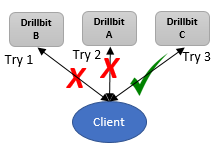
jdbc:drill:drillbit=<node name>[:<port>][,<node name2>[:<port>]...

<directory>/<cluster ID>;[schema=<storage plugin>];tries=5

“试”选项表示客户端可以尝试建立成功连接的唯一钻取位的最大数量。默认值为5.此选项首先尝试连接钻头时会改善Drill客户端的容错能力，钻头将作为Foreman（驱动查询的节点）。

客户端尝试连接到drillbits的顺序可能不会按连接字符串中列出的顺序发生。如果第一次尝试导致身份验证失败，则客户端不会尝试任何其他尝试。如果drillbit参数中列出的唯一钻头数小于“试图”值，客户端将尝试连接到每个钻头一次。

例如，如果在连接字符串中列出了三个唯一的drillbits，并且“tries”值设置为5，则客户端可以尝试连接到每个drillbit，直到连接成功为止，如下图所示：



如果客户端无法成功连接到任何drillbits，Drill将返回一条失败消息。

有关其他URL组件的定义，请参阅[将JDBC URL用于随机钻取连接](http://drill.apache.org/docs/using-the-jdbc-driver/#using-the-jdbc-url-for-a-random-drillbit-connection)。

### 使用钻机驱动程序类名称

JDBC驱动程序的类名是[org.apache.drill.jdbc.Driver](http://drill.apache.org/api/1.2/jdbc/)。有关详细信息，请参阅Apache Drill JDBC驱动程序版本1.2.0 [Javadoc](http://drill.apache.org/api/1.2/jdbc/)。

### 以编程方式连接到钻取的示例

以下示例代码显示了在使用Drill-Jdbc-all驱动程序连接到Drill的代码片段中使用类名称：

Class.forName("org.apache.drill.jdbc.Driver");

Connection connection =DriverManager.getConnection("jdbc:drill:zk=

node3.mynode.com:2181/drill/my\_cluster\_com-drillbits");

Statement st = connection.createStatement();

ResultSet rs = st.executeQuery("SELECT \* from cp.`employee`");

while(rs.next()){

System.out.println(rs.getString(1));

}

## 在Windows上使用JDBC和SQuirreL

### 2017年5月9日

要使用JDBC驱动程序通过SQuirreL访问Drill，请确保您满足先决条件并按照本节中的步骤操作。

先决条件

* SQuirreL需要JRE 7
* 在分布式模式下，在群集中的一个或多个节点上安装钻取。有关更多信息，请参阅[安装演练](http://drill.apache.org/docs/install-drill/)文档。
* 客户端必须能够使用IP解析Drill节点的实际主机名。验证是否在客户机上为Drill节点创建了一个DNS条目。

如果DNS条目不存在，请为“钻取”节点创建条目。

\* For Windows, create the entry in the %WINDIR%\system32\drivers\etc\hosts file.

\* For Linux and Mac, create the entry in /etc/hosts.

例： 127.0.1.1 maprdemo

第1步：获取钻取JDBC驱动程序

钻取JDBC驱动程序JAR文件必须存在于Windows计算机上的目录中，以便在SQuirreL客户端中配置驱动程序。

您可以将Drill JDBC JAR文件从安装了Drill的节点上的以下Drill安装目录复制到Windows机器上的目录中：

<drill\_installation\_directory>/jars/jdbc-driver/drill-jdbc-all-<version>.jar

或者，[将最新的Drill版本](http://apache.osuosl.org/drill/)的[TAR文件](http://apache.osuosl.org/drill/)下载到Windows计算机上的某个位置，并提取该文件的内容。您可能需要使用解压缩实用程序（如[7-zip）](http://www.7-zip.org/)来提取归档文件。解压缩后，可以在以下目录中找到驱动程序：

<windows\_directory>\apache-drill-<version>\jars\jdbc-driver\drill-jdbc-all-<version>.jar

第2步：安装和启动SQuirreL

要安装并启动SQuirreL，请完成以下步骤：

1. 从以下位置下载适用于Windows的SQuirreL JAR文件：[http](http://www.squirrelsql.org/#installation) :   
   [//www.squirrelsql.org/#installation](http://www.squirrelsql.org/#installation)
2. 双击SQuirreL JAR文件。SQuirreL安装向导将引导您完成安装过程。
3. 安装完成后，导航至SQuirreL安装文件夹，然后双击squirrel-sql.bat以启动SQuirreL。

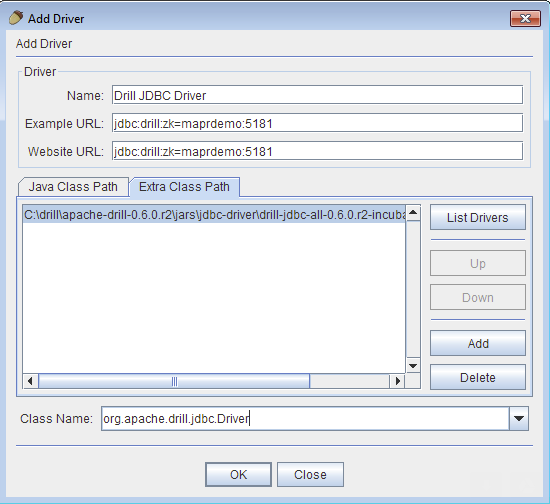
第3步：将钻取JDBC驱动程序添加到SQuirreL

要将Drill JDBC驱动程序添加到SQuirreL，请定义驱动程序并创建数据库别名。别名是驱动程序配置的特定实例。SQuirreL使用驱动程序定义和别名连接到Drill，以便访问您使用Drill注册的数据源。

**A.定义驱动程序**

要定义Drill JDBC驱动程序，请完成以下步骤：

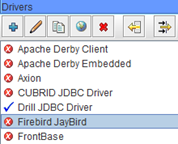
1. 在SQuirreL工具栏中，选择**Drivers> New Driver**。出现添加驱动程序对话框。



1. 输入以下信息：

| **选项** | **描述** |
| --- | --- |
| 名称 | 钻取JDBC驱动程序的名称 |
| 示例网址 | JDBC：钻头：ZK =[;模式=]示例：jdbc：drill：zk = maprdemo：5181注意：默认的ZooKeeper端口是2181.在MapR群集中，ZooKeeper端口是5181。 |
| 网址 | JDBC：钻头：ZK =[;模式=]示例：jdbc：drill：zk = maprdemo：5181注意：默认的ZooKeeper端口是2181.在MapR群集中，ZooKeeper端口是5181。 |
| 额外类路径 | 单击添加并导航到Windows目录中的JDBC JAR文件位置：/jars/jdbc-driver/drill-jdbc-all-0.6.0-incubating.jar选择JAR文件，单击“打开”，然后单击“列出驱动程序”。 |
| 班级名称 | 从下拉菜单中选择org.apache.drill.jdbc.Driver。 |

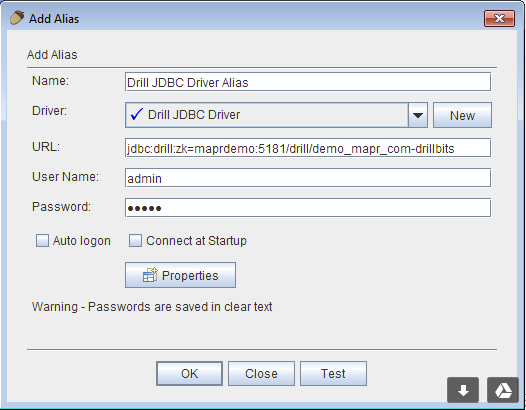
1. 点击**OK**。SQuirreL客户端显示一条消息，说明驱动程序注册成功，您可以在Drivers面板中看到驱动程序。



**B.创建一个别名**

要创建别名，请完成以下步骤：

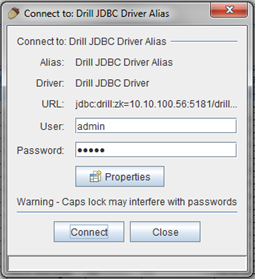
1. 选择**别名**选项卡。
2. 在SQuirreL工具栏中，选择**别名> 新别名**。出现添加别名对话框。



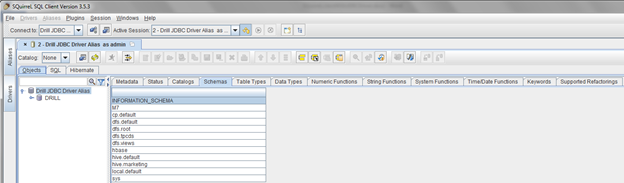
1. 输入以下信息：

* 别名：Drill JDBC Driver别名的唯一名称
* 驱动程序：选择钻取JDBC驱动程序
* URL：使用ZooKeeper中存储的Drill目录的名称和集群ID输入连接URL，如[下一节](http://drill.apache.org/docs/using-jdbc-with-squirrel-on-windows/#entering-the-connection-url)所示。
* 用户名：admin
* 密码：admin

1. 点击**确定**。出现连接到：对话框。



1. 点击**连接**。SQuirreL显示一条消息，指出连接成功。



1. 点击**OK**。SQuirreL显示一系列选项卡。

**输入连接URL**

在创建别名的过程的第3步中，使用以下语法输入包含存储在ZooKeeper中的Drill目录的名称和集群ID的连接URL：

jdbc:drill:zk=<zookeeper\_quorum>/<drill\_directory\_in\_zookeeper>/<cluster\_ID>;schema=<schema\_to\_use\_as\_default>

以下示例显示了安装在单个节点上的Drill的URL：

jdbc:drill:zk=10.10.100.56:5181/drill/demo\_mapr\_com-drillbits;schema=hive

jdbc:drill:zk=10.10.100.24:2181/drill/drillbits1;schema=hive

以下示例显示了以分布式模式安装并连接到ZooKeeper法定人数的Drill的URL：

jdbc:drill:zk=10.10.100.30:5181,10.10.100.31:5181,10.10.100.32:5181/drill/drillbits1;schema=hive

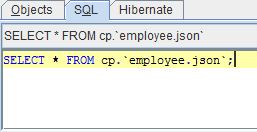
* 包括默认模式是可选的。
* ZooKeeper端口为2181.在MapR群集中，ZooKeeper端口为5181。
* 存储在ZooKeeper中的Drill目录是/drill。
* 钻取默认群集ID是drillbits1。

第4步：从SQuirreL运行查询

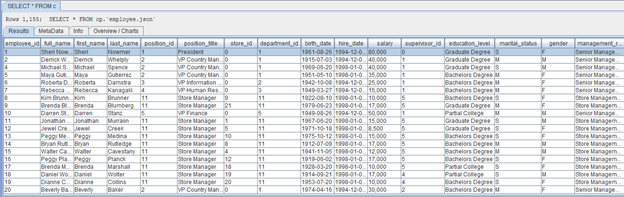
一旦通过Drill JDBC Driver将SQuirreL成功连接到集群，就可以从SQuirreL客户端发出查询。您可以对Drill安装中包含的一些示例数据运行测试查询，以使用Drill来试用SQuirreL。

要用Squirrel查询示例数据，请完成以下步骤：

1. 单击SQL选项卡。
2. 在查询框中输入以下查询：
3. SELECT \* FROM cp.`employee.json`;

例：  


1. 按**Ctrl + Enter**运行查询。以下查询结果显示：



您已成功运行SQuirreL客户端的查询。

## Dirll 资源管理器 Drill Exploer（类似Navicat）

### 简介

Drill Explorer是用于浏览Drill数据源，预览SQL查询结果以及创建视图的用户界面。通常，您可以使用Drill Explorer浏览数据，或者创建一个可以像查询表一样查询的视图。例如，在使用BI报告工具设计报告之前，使用Drill Explorer快速熟悉数据。在兼容ODBC的BI工具中，使用ODBC DSN创建ODBC连接。

您可以将Drill连接到Hive数据源，使用Drill Explorer创建一个视图，并像视图一样连接到视图。同样，您可以将Drill连接到HBase，Parquet，JSON，CSV或TSV文件，创建一个视图，并将该视图当作表来查询。

### 数据源连接：（使用）

2017年6月6日

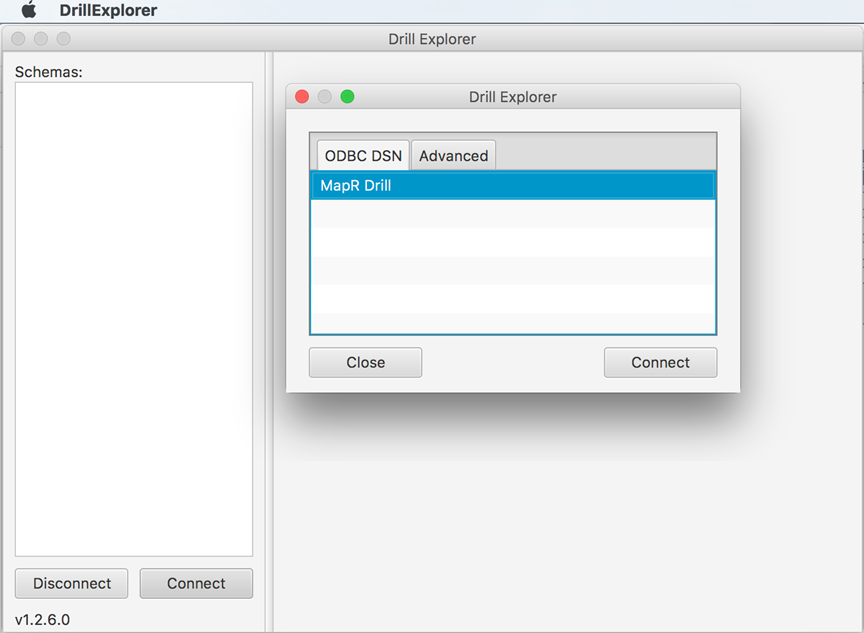
您启动Drill Explorer的方式因平台而异，但无论平台如何，Drill必须正在运行。

#### 将资源管理器连接到Linux上的数据

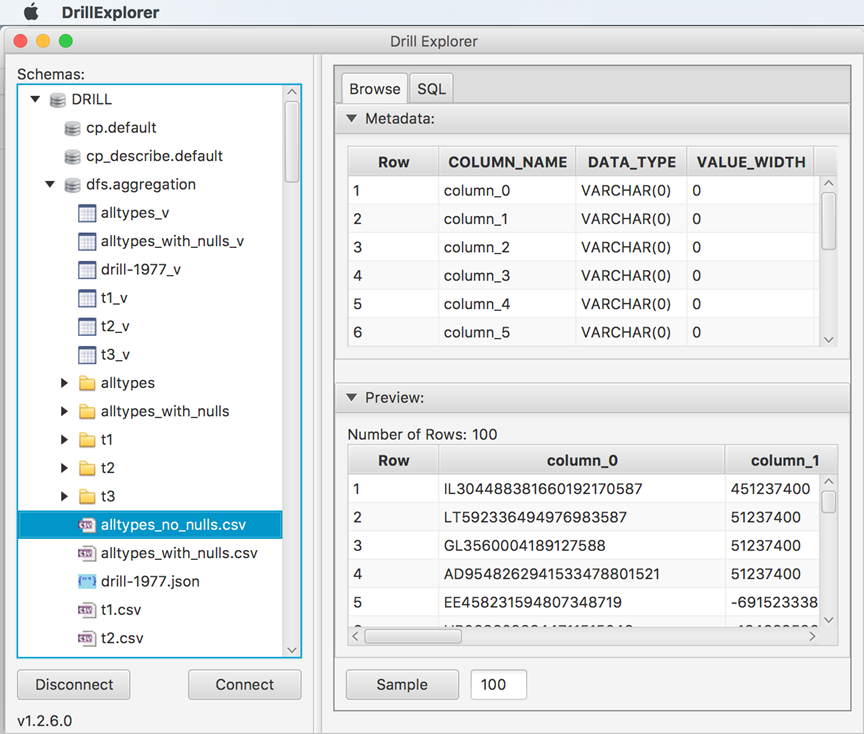
您需要X-11 XDisplay才能在Linux上使用Drill Explorer。运行DrillExplorer可执行文件/opt/mapr/drillodbc/DrillExplorer，然后按照下一节中的步骤2的说明，将Drill Explorer连接到Mac OS X上的数据。

#### 将资源管理器连接到Mac OS X上的数据

1. 在正在运行Drill的节点上，运行Drill Explorer。Drill Explorer应用程序位于/Applications目录中。出现Drill Explorer控制台。
2. 在控制台上单击**连接**。钻取资源管理器对话框出现。



1. 如果通过DSN连接，请选择在“ **ODBC DSN”**选项卡上配置的DSN的名称。例如，选择**MapR Drill**。或者，如果使用无DSN连接，请在“ **高级”**选项卡的文本框中键入连接字符串。例如，键入以下连接字符串：
2. DRIVER=MapR Drill ODBC Driver;AdvancedProperties={HandshakeTimeout=0;QueryTimeout=0;TimestampTZDisplayTimezone=utc;ExcludedSchemas=sys, INFORMATION\_SCHEMA;};Catalog=DRILL;Schema=;ConnectionType=ZooKeeper;ZKQuorum=192.168.39.43:5181;ZKClusterID=drillbits1
3. 如果将身份验证类型属性设置为启用模拟的**普通**（或基本身份验证），请回复提示输入用户名和密码。
4. 点击对话框中的**连接**。在“ **模式”**面板中，将显示连接的数据源的模式。



#### 在Windows上启动Drill Explorer

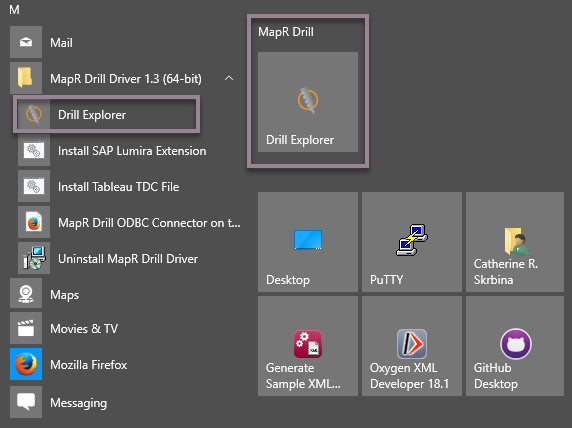
在Windows 10中，可以通过两种方式启动Drill Explorer：

* 开始菜单
* ODBC数据源管理员

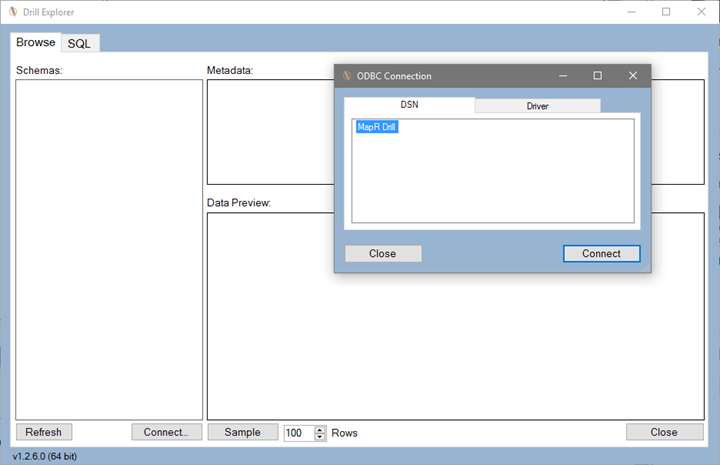
##### 开始菜单

从开始菜单启动Drill Explorer：

1. 点击**开始**，然后找到**MaprR Drill Driver 1.3 < 版本 >**。
2. 点击**钻取资源管理器**。



“ 钻取资源管理器”中将显示“显示**DSN”**选项卡的“ **ODBC连接”**对话框。

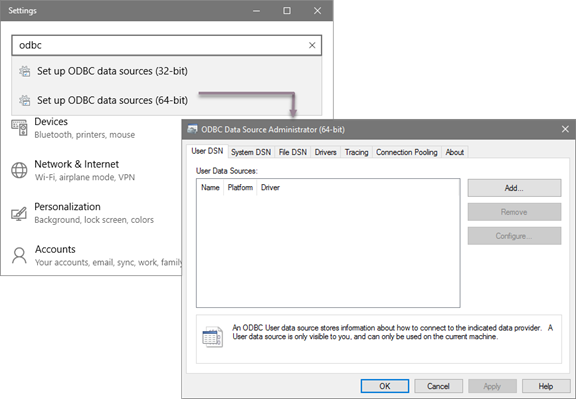


3.选择您想要浏览的DSN。例如，选择**MapR Drill**并单击**连接**。

##### ODBC数据源管理员

从ODBC数据源管理器启动Drill Explorer：

1. 输入**odbc**并从“Windows设置”字段中选择该实用程序的一个版本。出现“ **ODBC数据源管理员< 版本 >”**对话框。



1. 单击 **系统DSN**选项卡。
2. 选择**MapR Drill**并单击**配置**。出现**MapR Drill ODBC驱动程序DSN设置**对话框，显示多个配置属性。

### 浏览数据

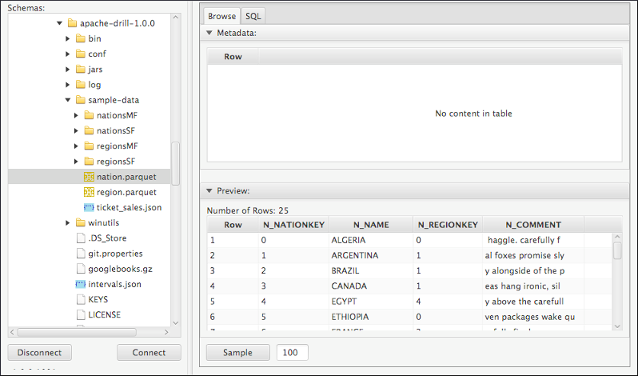
如果您有权读取文件和目录，则可以浏览它们。在“架构”窗格中，展开分支，直到找到要查看的文件或表格，然后选择它。

提示：要避免可以忽略的错误消息，请在查找有效数据之前，通过单击箭头而不是目录名称来仔细展开目录。

在“浏览”选项卡中，“元数据”窗格显示所选表或文件的结构（如果有）。预览窗格显示实际的数据。“SQL”选项卡显示返回在“浏览”选项卡中看到的数据的语句。

要在数据源中进行更改后更新数据，请在“模式”窗格中右键单击文件，表格，视图，架构或目录，然后选择刷新。更新范围仅限于您右键单击的项目。

以下示例显示如何使用示例MapR ODBC驱动程序DSN浏览数据。

1. 如有必要，开始钻取。
2. 启动Drill Explorer并连接到示例DSN。
3. 在“ 钻取资源管理器” 的“ **架构”**窗格中的“ **浏览”**选项卡上，导航到sample-datadfs.default架构中的目录。点击nation.parquet。  
   数据来自nation.parquet“预览”窗格。

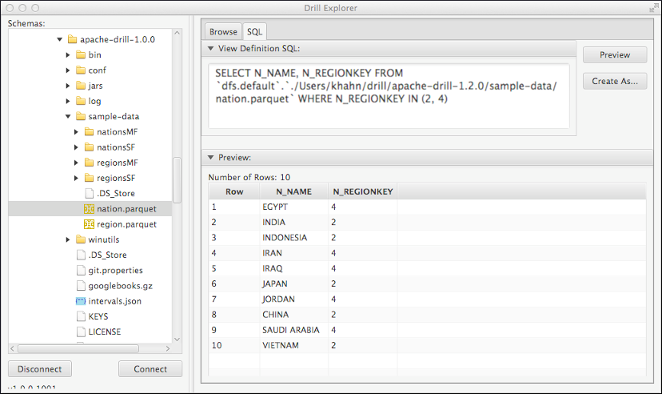
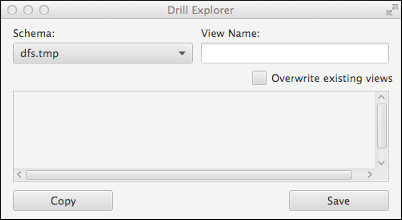
#### 定义一个视图

浏览数据后，您可以定义，预览和保存基于文件的数据的视图。在“SQL”选项卡上，修改“查看定义SQL”中的语句以选择要查看的数据。单击“ **创建**为”导航到保存视图的位置。您需要将视图保存到您writable在存储插件定义中定义的位置。保存视图后，会显示有关操作状态的消息。单击**复制**以复制该消息。

以下示例定义并保存了几列nation.parquet数据的视图。

1. 开始练习。
2. 启动Drill Explorer并连接到示例DSN。
3. 在**架构**上部分**浏览**选项卡，浏览到/sample-data/nation.parquet的dfs.default架构。
4. 在“ **SQL”**选项卡的“ **查看定义SQL”**字段中，输入查询，如“ [指定列名称”中所述](http://drill.apache.org/docs/browsing-data-and-defining-views/)。例如，修改SELECT语句如下：
5. SELECT N\_NAME, N\_REGIONKEY FROM `dfs.default`.`./Users/drilluser/drill/sample-data/nation.parquet` WHERE N\_REGIONKEY IN (2, 4)

不要在SQL语句的末尾放一个分号。

1. 点击**预览**。
2. 点击**创建为**。  
   “ 创建为”对话框显示。
3. 从模式列表中选择定义保存视图位置的模式。例如，接受[dfs.tmp](http://drill.apache.org/docs/query-directory-functions/#query-directory-function-example)模式，因为默认dfs存储插件将tmp目录定义为可写。您只能将视图保存到基于文件的模式。
4. 在**视图名称**中输入**视图的名称**。  
   不要在视图名称中包含空格。
5. 单击**保存，**然后选中**覆盖现有视图，**如果要保存现有视图。  
   相关的状态信息出现。
6. 点击**关闭**。

使用Drill Explorer创建的视图不会显示在与数据源类型关联的模式下。从保存视图时选择的基于文件的架构访问视图。

#### 指定列名称

使用SQL星号（\*）选择器来返回表中的所有列是有限的。Tableau可能无法使用Drill Explorer创建的SQL视图成功连接到您的Drill数据源。根据模式类型或文件格式，修改由Drill Explorer生成的查询以根据以下语法指导指定要返回的列。

在Drill Explorer中定义视图时指定单个列，而不是尝试全选，如以下示例所示。

SELECT CAST(account['name'] AS varchar(20)) FROM hbase.students

SELECT CAST(column1 AS varchar(20)) FROM `dfs`.`default`.`./opt/drill/test.parquet`

SELECT column1 FROM `dfs`.`default`.`./opt/drill/interval.json`

要查询嵌套元素，请使用以下语法，其中menu是column1的子元素：

SELECT column1['menu'] FROM `dfs`.`default`.`./opt/drill/interval.json`

您可以查询多个层次的元素。继续这个例子，如果menuitem是菜单的子项，那么使用下面的语法：

SELECT column1['menu']['menuitem'] FROM `dfs`.`default`.`./opt/drill/interval.json`

#### 投射数据要求

在SQL语句中，您需要明确地将二进制数据转换为另一种格式来查看数据。例如，以下查询以二进制格式显示HBase数据库的结果：  
SELECT account['name'] FROM hbase.students

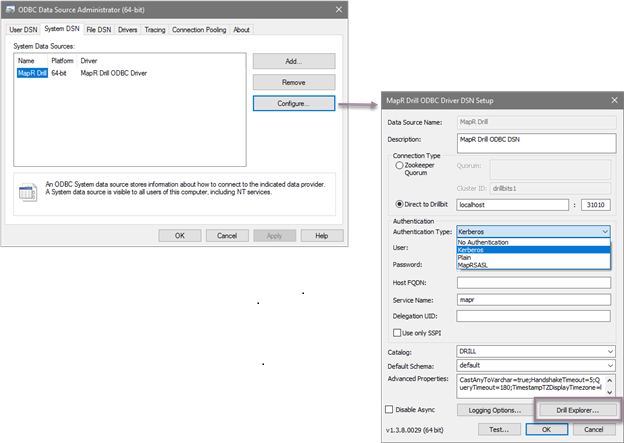
以下查询以字符串格式显示相同的结果：  
SELECT CAST(account['name'] AS varchar(20)) FROM hbase.students

以下查询以二进制格式显示来自Parquet文件的结果：  
SELECT column1 FROM `dfs`.`default`.`./opt/drill/test.parquet`

以下查询以字符串格式显示相同的结果：

SELECT CAST(column1 AS varchar(20)) FROM `dfs`.`default`.`./opt/drill/test.parquet`

您也可以将数据转换为其他数据类型，如整数或日期格式。



1. 点击对话框底部的**Drill Explorer**。出现Drill Explorer。

# 开发自定义函数

* [开发自定义函数介绍](http://drill.apache.org/docs/develop-custom-functions-introduction/)
* [开发一个简单的功能](http://drill.apache.org/docs/developing-a-simple-function/)
* [教程：开发一个简单的功能](http://drill.apache.org/docs/tutorial-develop-a-simple-function/)
* [开发一个聚合函数](http://drill.apache.org/docs/developing-an-aggregate-function/)
* [将自定义函数添加到钻取](http://drill.apache.org/docs/adding-custom-functions-to-drill/)
  + [添加自定义函数钻取简介](http://drill.apache.org/docs/adding-custom-functions-to-drill-introduction/)
  + [手动添加自定义函数到钻取](http://drill.apache.org/docs/manually-adding-custom-functions-to-drill/)
  + [创建自定义身份验证器](http://drill.apache.org/docs/creating-custom-authenticators/)
  + [动态UDF](http://drill.apache.org/docs/dynamic-udfs/)
* [在查询中使用自定义函数](http://drill.apache.org/docs/using-custom-functions-in-queries/)
* [自定义函数接口](http://drill.apache.org/docs/custom-function-interfaces/)

## 开发自定义函数介绍

2016年11月21日

Drill提供了一个高性能的Java API接口，您可以使用它来开发简单的集合自定义函数。自定义函数是在Java中开发的可重用SQL函数，用于封装在查询期间处理列值的代码。

自定义函数从SQL语句中调用，就像常规函数一样，并返回一个值。自定义函数的功能类似于钻取原始操作。他们可以执行内置SQL运算符和函数不提供的计算和转换。

### 简单的功能

一个简单的函数在一行上运行，并产生一行作为输出。在查询中包含一个简单的函数时，函数将在结果集中的每一行被调用一次。数学和字符串函数是简单函数的例子。

您可以使用提供的[教程](http://drill.apache.org/docs/tutorial-develop-a-simple-function/)创建一个基于github项目的简单函数，您可以下载该项目。

### 集合函数

用于开发集合自定义函数的API处于alpha阶段，仅供实验使用。集合函数与它们作为输入接受的行数的简单函数不同。一个聚集函数在多个输入行上运行，并产生一个单行作为输出。

COUNT（），MAX（），SUM（）和AVG（）函数是聚合函数的示例。您可以在具有GROUP BY子句的查询中使用聚合函数来为GROUP BY子句中的每个值组合生成一个具有单独聚合值的结果集。

### 开发过程

要为Drill开发自定义函数，请创建一个实现Drill的[简单](http://drill.apache.org/docs/developing-a-simple-function/)或[集合](http://drill.apache.org/docs/developing-an-aggregate-function/)接口的Java程序，然后将您的自定义函数添加到Drill。

从Drill 1.9开始，有两种向Drill添加自定义函数的方法。管理员可以手动将自定义函数添加到Drill，或者用户可以发出CREATE FUNCTION USING JAR命令来注册他们的自定义函数。CREATE FUNCTION USING JAR命令是[Dynamic UDF功能的一部分](http://drill.apache.org/docs/dynamic-udfs/)，需要管理员的协助。

## 开发一个简单的功能

本节简要介绍开发一个简单的功能。下一节中的[教程](http://drill.apache.org/docs/tutorial-develop-a-simple-function/)将介绍如何更深入地创建一个简单的函数。

为了开发一个简单的函数，你需要在一个Java程序包中创建一个类，该程序包实现了Drill简单的接口，并包含函数类型所需的信息。你的函数必须使用Drill支持的[数据类型](http://drill.apache.org/docs/supported-data-types/)。请记住以下要求：

* 注释UDF使用的所有数据。
* 完全符合课程参考。

使用Drill的简单函数接口完成以下步骤来开发一个简单的函数：

1. 创建一个Maven项目并添加以下依赖项：
2. <dependency>
3. <groupId>org.apache.drill.exec</groupId>
4. <artifactId>drill-java-exec</artifactId>
5. <version>1.1.0</version>
6. </dependency>
7. 创建一个实现DrillSimpleFunc接口的类，并将其作为范围标识FunctionScope.SIMPLE。

**例**

@FunctionTemplate(name = "myaddints", scope = FunctionScope.SIMPLE, nulls = NullHandling.NULL\_IF\_NULL)

public static class IntIntAdd implements DrillSimpleFunc {

1. 提供在Param和Output位持有者的代码中使用的变量。

**例**

@Param IntHolder in1;

@Param IntHolder in2;

@Output IntHolder out;

1. 添加在eval（）方法中为函数执行操作的代码。

**例**

public void setup() {

}

public void eval() {

out.value = in1.value + in2.value;

}

1. 使用maven-source-plugin来编译源文件和类JAR文件。验证drill-module.conf资源文件夹中是否包含空白。  
   练习在类路径扫描期间搜索这个模块。如果文件未包含在资源文件夹中，则可以将其添加到JAR文件或将其添加到 etc/drill/conf。

## 教程：开发一个简单的功能

2016年11月18日

您可以创建具有高级功能的复杂功能，但出于示例目的，本教程将介绍如何创建简单的MASK功能。使用[Drill Simple Function界面](https://github.com/apache/drill/blob/master/exec/java-exec/src/main/java/org/apache/drill/exec/expr/DrillSimpleFunc.java)扩展Drill查询引擎：

package org**.**apache**.**drill**.**exec**.**expr**;**

import org.apache.drill.exec.record.RecordBatch**;**

**public** **interface** **DrillSimpleFunc** **extends** DrillFunc**{**

**public** **void** **setup();**

**public** **void** **eval();**

**}**

MASK函数转换每一行的列值。该函数将屏蔽字符串中的字符，如以下伪代码所示：

MASK( 'PASSWORD' , '#' , 4 ) => ####WORD

MASK函数用一个字符串的开头替换给定数量的字符与另一个字符。在这个例子中，MASK用＃字符替换四个字符。

您可以从[drill-simple-mask-function Github存储库中](https://github.com/tgrall/drill-simple-mask-function)获取用于创建和构建此函数的完整项目。

### 先决条件

* [Oracle Java SE开发（JDK）工具包7](http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html)或更高版本
* [Apache Drill 1.1](http://getdrill.org/drill/download/apache-drill-1.1.0.tar.gz)或更高版本
* [Maven 3.0](https://maven.apache.org/download.cgi)或更高版本

### 第1步：添加依赖关系

首先，将以下Drill依赖项添加到您的maven项目中：

<dependency>

<groupId>org.apache.drill.exec</groupId>

<artifactId>drill-java-exec</artifactId>

<version>1.1.0</version>

</dependency>

### 第2步：向功能模板添加注释

要开始实现DrillSimpleFunc接口，请将以下注释添加到@FunctionTemplate声明中：

* 自定义函数的名称 name="mask"
* 自定义函数的范围，在这种情况下，简单 scope= FunctionTemplate.FunctionScope.SIMPLE
* 值为NULL时该做什么，在这种情况下，Reverse将仅返回NULLnulls = FunctionTemplate.NullHandling.NULL\_IF\_NULL

**.** **.** **.**

package org**.**apache**.**drill**.**contrib**.**function**;**

import org.apache.drill.exec.expr.DrillSimpleFunc**;**

import org.apache.drill.exec.expr.annotations.FunctionTemplate**;**

@FunctionTemplate**(**

name**=**"mask"**,**

scope**=** FunctionTemplate**.**FunctionScope**.**SIMPLE**,**

nulls **=** FunctionTemplate**.**NullHandling**.**NULL\_IF\_NULL

**)**

**public** **class** **SimpleMaskFunc** **implements** DrillSimpleFunc**{**

**public** **void** **setup()** **{**

**}**

**public** **void** **eval()** **{**

**}**

**}**

### 步骤3：声明输入参数

这个函数将被动态的生成，就像你在[DrillSimpleFuncHolder中](https://github.com/apache/drill/blob/master/exec/java-exec/src/main/java/org/apache/drill/exec/expr/fn/DrillSimpleFuncHolder.java/#L42)看到的[一样](https://github.com/apache/drill/blob/master/exec/java-exec/src/main/java/org/apache/drill/exec/expr/fn/DrillSimpleFuncHolder.java/#L42)，输入参数和输出持有者是通过注解来定义的。使用@Param注释定义参数。

* 可空的字符串
* 掩码字符或字符串
* 从第一个开始替换的字符数

使用持有者类来提供一个缓冲区来高效地管理较大的对象：VarCharHolder或NullableVarCharHolder。

**.** **.** **.**

**public** **class** **SimpleMaskFunc** **implements** DrillSimpleFunc **{**

@Param

NullableVarCharHolder input**;**

@Param**(**constant **=** **true)**

VarCharHolder mask**;**

@Param**(**constant **=** **true)**

IntHolder toReplace**;**

**.** **.** **.**

**}**

**注意**

Drill实际上并没有在查询中使用Java堆来处理数据，而是将这些数据保存在堆中，并在不使用Java垃圾收集器的情况下管理我们的生命周期。

### 步骤4：声明返回值类型

另外，使用@Output注释，将返回值定义为VarCharHolder类型。因为你正在操作一个VarChar，你还必须注入一个Drill用于输出的缓冲区。

**public** **class** **SimpleMaskFunc** **implements** DrillSimpleFunc **{**

**.** **.** **.**

@Output

VarCharHolder out**;**

@Inject

DrillBuf buffer**;**

**.** **.** **.**

**}**

### 第5步：实现eval（）方法

MASK功能不需要任何设置，所以你不需要定义setup（）方法。只定义eval（）方法。

**public** **void** **eval()** **{**

*// get the value and replace with*

String maskValue **=** org**.**apache**.**drill**.**exec**.**expr**.**fn**.**impl**.**StringFunctionHelpers**.**getStringFromVarCharHolder**(**mask**);**

String stringValue **=** org**.**apache**.**drill**.**exec**.**expr**.**fn**.**impl**.**StringFunctionHelpers**.**toStringFromUTF8**(**input**.**start**,** input**.**end**,** input**.**buffer**);**

**int** numberOfCharToReplace **=** Math**.**min**(**toReplace**.**value**,** stringValue**.**length**());**

*// build the mask substring*

String maskSubString **=** com**.**google**.**common**.**base**.**Strings**.**repeat**(**maskValue**,** numberOfCharToReplace**);**

String outputValue **=** **(new** **StringBuilder(**maskSubString**)).**append**(**stringValue**.**substring**(**numberOfCharToReplace**)).**toString**();**

*// put the output value in the out buffer*

out**.**buffer **=** buffer**;**

out**.**start **=** 0**;**

out**.**end **=** outputValue**.**getBytes**().**length**;**

buffer**.**setBytes**(**0**,** outputValue**.**getBytes**());**

**}**

eval（）方法执行以下任务：

* 获取面具
* 获取值
* 获取要替换的字符数
* 用掩码值生成一个新的字符串
* 创建并填充输出缓冲区

即使是经验丰富的Java开发人员，eval（）方法也可能看起来有点奇怪，因为Drill会即时生成最终代码以满足查询请求。该技术利用Java的即时（JIT）编译器实现最高速度。

基本编码规则

为了充分利用Java的即时（JIT）编译器，您需要遵循一些基本规则。

* 不要使用导入。相反，请使用Apache Drill中打包的Google Guava API所要求的全限定类名，如[“步骤3：声明输入参数”所示](http://drill.apache.org/docs/tutorial-develop-a-simple-function/#step-3:-declare-input-parameters)。
* 通过调用帮助器方法（如getStringFromVarCharHolder和toStringFromUTF8）来操作ValueHolders类，例如VarCharHolder和IntHolder，如[“步骤5：实现eval（）函数”所示](http://drill.apache.org/docs/tutorial-develop-a-simple-function/#step-5:-implement-the-eval()-method)。
* 不要调用诸如toString之类的方法，因为这会导致严重的问题。

### 完整的代码清单

Github钻取简单掩码函数项目包含MASK函数代码的[完整列表](https://github.com/tgrall/drill-simple-mask-function/blob/master/src/main/java/org/apache/drill/contrib/function/SimpleMaskFunc.java)。

### 准备包裹

由于Drill生成源代码，因此必须准备程序包，以使函数的类和源出现在类路径中。必要的代码生成需要类和源。Drill使用已编译的代码来访问注释，并使用源代码来执行代码生成。这个包装不同于Java代码通常打包的方式。

为了简单起见，请使用maven在pom.xml文件中构建您的项目，如下所示：

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-source-plugin</artifactId>

<version>2.4</version>

<executions>

<execution>

<id>attach-sources</id>

<phase>package</phase>

<goals>

<goal>jar-no-fork</goal>

</goals>

</execution>

</executions>

</plugin>

将drill-module.conf文件添加到资源

drill-module.conf在项目的资源文件夹中添加一个文件。这个文件的存在告诉Drill你的jar包含一个自定义函数。把下面的行放在drill-module.config：

drill.classpath.scanning.packages += "org.apache.drill.contrib.function"

构建和部署功能

使用mvn包构建函数：

mvn clean package

Maven生成两个JAR文件：

* 带有类和资源的默认jar（drill-simple-mask-1.0.jar）
* 第二个jar文件（drill-simple-mask-1.0-sources.jar）

将JAR文件添加到Drill，方法是将其复制到以下位置：

<Drill installation directory>/jars/3rdparty

**注意：**本教程显示了将JAR文件添加到Drill的手动方法，但是从Drill 1.9开始，Dynamic UDF功能为用户提供了一种新方法。

测试新功能

重新启动钻取并[employee.json](http://drill.apache.org/docs/querying-json-files/)使用Drill安装的文件上运行以下查询：

SELECT MASK(first\_name, '\*' , 3) FIRST , MASK(last\_name, '#', 7) LAST FROM cp.`employee.json` LIMIT 5;

+----------+------------+

| FIRST | LAST |

+----------+------------+

| \*\*\*ri | ###### |

| \*\*\*rick | ####### |

| \*\*\*hael | ###### |

| \*\*\*a | #######ez |

| \*\*\*erta | ####### |

+----------+------------+

5 rows selected (2.259 seconds)

## 将自定义函数添加到钻取

* [添加自定义函数钻取简介](http://drill.apache.org/docs/adding-custom-functions-to-drill-introduction/)
* [手动添加自定义函数到钻取](http://drill.apache.org/docs/manually-adding-custom-functions-to-drill/)
* [创建自定义身份验证器](http://drill.apache.org/docs/creating-custom-authenticators/)
* [动态UDF](http://drill.apache.org/docs/dynamic-udfs/)

### 添加自定义函数钻取简介

2016年11月22日

从Drill 1.9开始，有两种向Drill添加自定义函数的方法。管理员可以手动向Drill添加函数，也可以为用户提供对临时目录的访问权限，他们可以使用CREATE FUNCTION USING JAR命令上传JAR文件并注册他们的UDF。CREATE FUNCTION USING JAR命令是Dynamic UDF功能的一部分。

* 有关手动指令，请参阅[手动将自定义功能添加到钻取](http://drill.apache.org/docs/manually-adding-custom-functions-to-drill/)。
* 有关动态UDF信息和说明，请参阅[动态UDF](http://drill.apache.org/docs/dynamic-udfs/)。

### 手动添加自定义函数到钻取

管理员可以手动将自定义函数添加到Drill。在定制函数开发之后，生成源文件和类JAR文件。将两个JAR文件添加到每个节点上的Drill类路径，并将包含这些类的包的名称包含到主Drill配置文件中。重新启动每个节点上的钻头以刷新配置。

要将自定义函数添加到“钻取”，请完成以下步骤：

1. 将自定义函数的源代码和类JAR文件通过复制到所有drillbits的Drill classpath中<drill installation directory>/jars/3rdparty。
2. drill-module.conf在类的JAR文件中包含一个文件，它的根。
3. 将以下代码添加到drill-module.conf（src / main / resources / drill-module.conf）中，并替换com.yourgroupidentifier.udf为您的UDF的包名称，如下所示：
4. drill.classpath.scanning.packages += "com.yourgroupidentifier.udf"

**注意：**用逗号分隔软件包名称。

1. 验证DRILL\_HOME/conf/drill-override.conf不包含有关UDF包的任何信息。
2. 发出以下命令以重新启动钻取：
3. <drill\_installation\_directory>/bin/drillbit.sh restart

现在，您可以在钻取查询中使用自定义函数。

### 动态UDF

2016年11月22日

Drill 1.9引入了对动态UDF的支持。动态UDF功能允许用户使用JAR和DROP FUNCTION USING JAR命令使用CREATE FUNCTION USING JAR和DROP FUNCTION USING JAR命令自行注册和取消注册UDF。

当管理员在多租户环境中手动加载和卸载UDF时，动态UDF功能消除了重新启动drillbits的需要，这可能会中断用户。用户可以发出CREATE FUNCTION USING JAR命令来注册手动加载的（内置的）UDF。另外，用户可以将注册的UDF迁移到内置的UDF。

动态UDF功能是默认启用的。管理员可以使用ALTER SYSTEM SET命令启用或禁用该功能exec.udf.enable\_dynamic\_support option。启用此功能后，用户必须先将UDF（源和二进制）JAR文件上载到分布式文件系统中的临时目录中，然后再发出CREATE FUNCTION USING JAR命令来注册UDF。

如果用户没有对分段目录的写入权限，则注册尝试失败。当用户发出CREATE FUNCTION USING JAR命令来注册UDF时，Drill在验证和注册UDF时使用特定的目录。在[持续配置存储](http://drill.apache.org/docs/persistent-configuration-storage/)存储UDF和相关的JAR文件的列表。Drills在注册和注销UDF时引用这个列表。

#### UDF目录

Drill在注册UDF时使用的目录drill.exec.udf在drill-override.conf文件的节中配置。在启动时，Drill将验证这些目录是否存在于文件系统中。如果这些目录不存在，Drill会创建它们。如果Drill无法创建目录，则启动尝试失败。管理员可以修改目录位置drill-override.conf。

配置文件包含使用Dynamic UDF功能所需的以下默认属性和目录：

drill.exec.udf: {

retry-attempts: 5,

directory: {

base: ${drill.exec.zk.root}"/udf",

local: ${drill.exec.udf.directory.base}"/local",

staging: ${drill.exec.udf.directory.base}"/staging",

registry: ${drill.exec.udf.directory.base}"/registry",

tmp: ${drill.exec.udf.directory.base}"/tmp"

}

}

下表介绍了配置属性和UDF目录，其中drill.exec.udf.directory.base是用于生成所有UDF目录（本地和远程）的相对目录：

| **属性** | **描述** |
| --- | --- |
| 重试尝试 | Drill返回错误之前，UDF注册表更新失败的次数。Drill在更新远程函数注册表之前检查注册表版本，以避免重写其他用户所做的更改。如果注册表版本已更改，则Drill将再次验证更新的注册表中的功能。默认值是5。 |
| base：$ {drill.exec.zk.root}“/ udf” | 该属性用于在使用相同文件系统的群集之间分隔UDF目录。 |
| local：$ {drill.exec.udf.directory.base}“/ local” | 相对路径连接到Drill临时目录以指示本地UDF目录。本地UDF目录用作动态UDF JAR文件的临时目录。Drill在退出时清除该目录。 |
| 暂存：$ {drill.exec.udf.directory.base}“/暂存” | 用户将其二进制和源JAR文件复制到的位置。这个目录必须是用户可以访问的，以便注册他们的UDF。当UDF注册时，Drill从这个目录中删除两个JAR文件（源文件和二进制文件）。如果Drill无法从JAR文件注册UDF，则JAR文件将保留在此处。你可以改变这个目录的位置。 |
| 注册表：$ {drill.exec.udf.directory.base}“/ registry” | Drill在验证UDF之后复制源和二进制JAR文件的位置。Drill将JAR文件从注册表目录复制到每个钻头的本地UDF目录。当您取消注册UDF时，Drill将从每个钻头的本地UDF目录中删除相应的JAR文件。不要从注册表目录中删除JAR文件。从注册表目录中删除JAR文件会导致Dynamic UDF注册表中指向存储JAR文件的目录中的不一致。你可以改变这个目录的位置。 |
| tmp：$ {drill.exec.udf.directory.base}“/ tmp” | 在开始注册过程之前，Drill将备份二进制和源JAR文件的位置。钻取将每个二进制文件和源文件放在此目录中的唯一文件夹中。在注册结束时，Drill从这个目录中删除这两个JAR文件。你可以改变这个目录的位置。 |

下表列出了可以添加的可选目录：

| **属性** | **描述** |
| --- | --- |
| drill.exec.udf.directory.fs | 从默认更改文件系统。如果群集中有多个drillbits，并且默认文件系统未分发，则必须包含此属性，并将其设置为分布式文件系统。例如，file：///，hdfs：///或maprfs：///，如下所示：drill.exec.udf.directory.fs：“hdfs：///” |
| drill.exec.udf.directory.root | 更改远程UDF目录的根目录。默认情况下，此属性设置为启动钻头的用户的主目录。例如，在Linux上，位置是/ home / some\_user。在DFS上，位置是/ user /。而且，在Windows上，位置是/ C：/ User /。 |

#### 安全和身份验证影响

目前，如果用户可以访问登台目录，则任何用户都可以注册UDF。由于Drill不提供完整的授权和身份验证支持，因此管理员可能希望禁用Dynamic UDF功能。请参阅[启用和禁用动态UDF功能](http://drill.apache.org/docs/dynamic-udfs/#enabling-and-disabling-the-dynamic-udf-feature)。

Drill将所有JAR文件从staging目录移动到其他UDF目录，而不是以提交JAR文件的用户身份作为启动drillbit的用户。即使启用了模拟，演练也是如此。

#### 使用动态UDF功能之前

在用户可以使用Dynamic UDF功能成功注册和取消注册其UDF之前，管理员和用户必须完成某些先决条件。下面列出了每个角色的先决条件。

#### 管理员

在用户使用JAR命令或使用JAR的DROP FUNCTION命令注册或取消注册UDF之前，用户可以发出CREATE FUNCTION，确认该exec.udf.enable\_dynamic\_support选项已启用，并且用户可以访问临时目录。请参阅[启用和禁用动态UDF功能](http://drill.apache.org/docs/dynamic-udfs/#enabling-and-disabling-the-dynamic-udf-feature)。

#### 用户

使用Drill的[简单](http://drill.apache.org/docs/developing-a-simple-function/)或[集合](http://drill.apache.org/docs/developing-an-aggregate-function/)函数接口创建一个UDF 。将drill-module.conf文件添加到类JAR文件的根目录。该drill-module.conf文件应包含要扫描功能的软件包drill.classpath.scanning.packages+= "com.mydomain.drill.fn"，如以下示例所示：

drill.classpath.scanning.package+= "com.mydomain.drill.fn"

替换com.mydomain.drill.fn为您的UDF的包名称。如果有多个软件包，请用逗号分隔软件包名称。

一旦创建了UDF，将源和二进制JAR文件复制到登台目录。如果您不知道暂存目录的位置，请联系您的管理员。现在，您可以使用CREATE FUNCTION USING JAR命令来注册您的UDF。请参阅[注册UDF](http://drill.apache.org/docs/dynamic-udfs/#registering-a-udf)。

#### 启用和禁用动态UDF功能

管理员可以启用或禁用Dynamic UDF功能。该功能默认启用。该exec.udf.enable\_dynamic\_support选项打开和关闭动态UDF功能。如果担心安全问题，管理员可以禁用该功能以防止用户注册和取消注册UDF。

使用带系统选项的[ALTER SYSTEM SET](http://drill.apache.org/docs/alter-system/)命令 exec.udf.enable\_dynamic\_support打开或关闭该功能。

#### 注册一个UDF

将UDF源文件和二进制JAR文件复制到DFS分段目录，然后发出CREATE FUNCTION USING JAR命令来注册UDF，如下所示：

CREATE FUNCTION USING JAR ‘<jar\_name>.jar’

如果您不知道登台目录的位置，或者需要访问目录，请联系管理员。

发出命令时，Drill使用JAR文件名在Dynamic UDF注册表（[持久性存储](http://drill.apache.org/docs/persistent-configuration-storage/)）中注册JAR名称，然后将源和二进制JAR文件复制到每个钻头的本地UDF目录中。

在成功注册后，Drill返回一个包含注册UDF列表的消息：

+---------------+-----------------------------------------------------------------------------------------------------------+

| ok | summary |

+---------------+-----------------------------------------------------------------------------------------------------------+

| true | The following UDFs in jar %s have been registered: %s |

+---------------+-----------------------------------------------------------------------------------------------------------+

#### 取消注册UDF

使用JAR命令发出DROP FUNCTION命令来取消注册UDF，如下所示：

DROP FUNCTION USING JAR ‘<jar\_name>.jar’

发出命令时，Drill根据JAR文件名取消注册UDF，并从UDF目录中删除JAR文件。Drill从UDF注册表中删除与JAR文件相关联的所有UDF（[持久存储](http://drill.apache.org/docs/persistent-configuration-storage/)，信令drillbits以开始本地取消注册过程。

Drill返回包含未注册UDF列表的消息：

+---------------+-----------------------------------------------------------------------------------------------------------+

| ok | summary |

+---------------+-----------------------------------------------------------------------------------------------------------+

| true | The following UDFs in jar %s have been unregistered: %s |

+---------------+-----------------------------------------------------------------------------------------------------------+

#### 将UDF从动态迁移到内置

您可以将使用Dynamic UDF功能注册的UDF迁移到内置UDF，以释放UDF目录和Dynamic UDF注册表（[持久性存储](http://drill.apache.org/docs/persistent-configuration-storage/)）中的空间。您可以迁移所有的UDF，也可以迁移一部分UDF。如果迁移所有UDF，则不能发出DROP FUNCTION USING JAR命令来注销已经从动态迁移到内置的UDF。

#### 将所有已注册的UDF迁移到内置的UDF

要将所有注册的UDF迁移到内置的UDF，请完成以下步骤：

1. 停止群集中的所有drillbits。
2. 将UDF源文件和二进制JAR文件移动到每个钻头的$ DRILL\_SITE / jars目录中。（必须包含在类路径中。）
3. 从ZooKeeper中删除远程函数注册表。
4. 启动集群中的所有drillbits。

#### 将一些注册的UDF JAR文件迁移到内置的UDF

要将部分UDF JAR文件迁移到内置UDF，请完成以下步骤：

1. 将JAR文件从UDF注册表目录复制（不移动）到每个钻头上的$ DRILL\_SITE / jars目录。（必须包含在类路径中。）
2. 对每个JAR文件发出DROP FUNCTION USING JAR命令。
3. 停止群集中的所有drillbits。
4. 启动集群中的所有drillbits。

#### 限制

动态UDF功能具有以下已知限制：

* 如果用户在引用UDF的查询正在运行时丢弃UDF，则查询可能失败。在发出DROP命令之前，用户应验证没有查询引用UDF。
* DROP命令仅在JAR级别名称下运行。用户不能从存在多个UDF的JAR中仅注销一个UDF。为了避免这种情况，用户可以为每个jar创建一个UDF。
* 即使动态UDF支持被禁用，所有UDF目录（远程或本地）也是在钻头启动时创建的。如果启动drillbit的用户没有对这些目录的写入权限，则Drillbit启动失败。

## 创建自定义身份验证器

2017年6月22日

管理员可以使用此处提供的模板来开发和实现基于自定义用户名/密码的身份验证器。

完成以下步骤来构建和实现自定义验证器：

1. 将以下Java文件构建到JAR文件中：
2. MyCustomDrillUserAuthenticatorImpl.java
3. package myorg.dept.drill.security;
4. import org.apache.drill.common.config.DrillConfig;
5. import org.apache.drill.exec.exception.DrillbitStartupException;
6. import org.apache.drill.exec.rpc.user.security.UserAuthenticator;
7. import org.apache.drill.exec.rpc.user.security.UserAuthenticationException;
8. import org.apache.drill.exec.rpc.user.security.UserAuthenticatorTemplate;
9. import java.io.IOException;
10. /\*
11. \* Implement {@link org.apache.drill.exec.rpc.user.security.UserAuthenticator} for illustrating how to develop a custom authenticator and use it in Drill
12. \*/
13. @UserAuthenticatorTemplate(type = "myCustomAuthenticatorType")
14. public class MyCustomDrillUserAuthenticatorImpl implements UserAuthenticator {
15. public static final String TEST\_USER\_1 = "testUser1";
16. public static final String TEST\_USER\_2 = "testUser2";
17. public static final String TEST\_USER\_1\_PASSWORD = "testUser1Password";
18. public static final String TEST\_USER\_2\_PASSWORD = "testUser2Password";
19. /\*\*
20. \* Setup for authenticating user credentials.
21. \*/
22. @Override
23. public void setup(DrillConfig drillConfig) throws DrillbitStartupException {
24. // If the authenticator has any setup such as making sure authenticator provider servers are up and running or
25. // needed libraries are available, it should be added here.
26. }
27. /\*\*
28. \* Authenticate the given <i>user</i> and <i>password</i> combination.
29. \*
30. \* @param userName
31. \* @param password
32. \* @throws UserAuthenticationException if authentication fails for given user and password.
33. \*/
34. @Override
35. public void authenticate(String userName, String password) throws UserAuthenticationException {
36. if (!(TEST\_USER\_1.equals(userName) && TEST\_USER\_1\_PASSWORD.equals(password)) &&
37. !(TEST\_USER\_2.equals(userName) && TEST\_USER\_2\_PASSWORD.equals(password))) {
38. throw new UserAuthenticationException("custom failure message if the admin wants to show it to user");
39. }
40. }
41. /\*\*
42. \* Close the authenticator. Used to release resources. Ex. LDAP authenticator opens connections to LDAP server,
43. \* such connections resources are released in a safe manner as part of close.
44. \*
45. \* @throws IOException
46. \*/
47. @Override
48. public void close() throws IOException {
49. // Any clean up such as releasing files/network resources should be done here
50. }
51. }
52. drill-module.conf使用以下配置代码创建一个名为的文件，然后将该文件添加到JAR文件的根目录中：
53. drill {
54. classpath.scanning {
55. packages += "myorg.dept.drill.security"
56. }
57. }

这使自定义类路径扫描程序能够找到新的类。

1. 将您构建的JAR文件添加到每个“钻取”节点上的以下目录中：  
   <DRILLINSTALL\_HOME>/jars
2. 将以下配置添加到位于以下位置drill.exec的drill-override.conf文件中的块<DRILLINSTALL\_HOME>/conf/：
3. drill.exec {
4. security.user.auth {
5. enabled: true,
6. packages += "myorg.dept.drill.security",
7. impl: "myCustomAuthenticatorType"
8. }
9. }
10. 在每个Drill节点上重新启动Drillbit进程。

<DRILLINSTALL\_HOME>/bin/drillbit.sh restart

用于开发集合自定义函数的API处于alpha阶段，仅供实验使用。为了试验这个API，在一个Java程序包中创建一个实现了Drill的聚集接口的类到程序中。包括功能所需的信息。您的函数必须包含Drill支持的数据类型，例如INTEGER或BIGINT。有关支持的数据类型的列表，请参阅[SQL参考](http://drill.apache.org/docs/supported-data-types/)。牢记以下准则：

* 不要使用复杂的@Workspace变量。
* 您不能分配重复的\*值或在@Workspace中有一个ComplexWriter。

完成以下步骤来创建一个聚合函数：

1. 创建一个Maven项目并添加以下依赖项：
2. <dependency>
3. <groupId>org.apache.drill.exec</groupId>
4. <artifactId>drill-java-exec</artifactId>
5. <version>1.1.0</version>
6. </dependency>
7. 创建一个实现DrillAggFunc接口的类，并将其作为范围标识FunctionTemplate.FunctionScope.POINT\_AGGREGATE。

**例**

@FunctionTemplate(name = "count", scope = FunctionTemplate.FunctionScope.POINT\_AGGREGATE)

public static class BitCount implements DrillAggFunc{

1. 提供在Param, Workspace,和Output位持有者的代码中使用的变量。

**例**

@Param BitHolder in;

@Workspace BitHolder value;

@Output BitHolder out;

1. 包括setup（），add（），output（），andreset（）方法。

**示例** public void setup（）{value = new BitHolder（）; value.value = 0; }

@Override

public void add() {

value.value++;

}

@Override

public void output() {

out.value = value.value;

}

@Override

public void reset() {

value.value = 0;

1. 使用maven-source-plugin来编译源文件和类JAR文件。验证drill-module.confJAR的资源文件夹中是否包含空白。  
   练习在类路径扫描期间搜索这个模块。如果文件未包含在资源文件夹中，则可以将其添加到JAR文件或将其添加到 etc/drill/conf。

## 开发一个聚合函数

用于开发集合自定义函数的API处于alpha阶段，仅供实验使用。为了试验这个API，在一个Java程序包中创建一个实现了Drill的聚集接口的类到程序中。包括功能所需的信息。您的函数必须包含Drill支持的数据类型，例如INTEGER或BIGINT。有关支持的数据类型的列表，请参阅[SQL参考](http://drill.apache.org/docs/supported-data-types/)。牢记以下准则：

* 不要使用复杂的@Workspace变量。
* 您不能分配重复的\*值或在@Workspace中有一个ComplexWriter。

完成以下步骤来创建一个聚合函数：

1. 创建一个Maven项目并添加以下依赖项：
2. <dependency>
3. <groupId>org.apache.drill.exec</groupId>
4. <artifactId>drill-java-exec</artifactId>
5. <version>1.1.0</version>
6. </dependency>
7. 创建一个实现DrillAggFunc接口的类，并将其作为范围标识FunctionTemplate.FunctionScope.POINT\_AGGREGATE。

**例**

@FunctionTemplate(name = "count", scope = FunctionTemplate.FunctionScope.POINT\_AGGREGATE)

public static class BitCount implements DrillAggFunc{

1. 提供在Param, Workspace,和Output位持有者的代码中使用的变量。

**例**

@Param BitHolder in;

@Workspace BitHolder value;

@Output BitHolder out;

1. 包括setup（），add（），output（），andreset（）方法。

**示例** public void setup（）{value = new BitHolder（）; value.value = 0; }

@Override

public void add() {

value.value++;

}

@Override

public void output() {

out.value = value.value;

}

@Override

public void reset() {

value.value = 0;

1. 使用maven-source-plugin来编译源文件和类JAR文件。验证drill-module.confJAR的资源文件夹中是否包含空白。  
   练习在类路径扫描期间搜索这个模块。如果文件未包含在资源文件夹中，则可以将其添加到JAR文件或将其添加到 etc/drill/conf。

## 在查询中使用自定义函数

当您向自定义函数发出查询时，Drill将在类路径中搜索与查询中的请求相匹配的函数。一旦Drill找到请求的功能，Drill将处理查询并在处理期间应用该功能。

您的Drill安装包含Drill类路径中的示例文件。一个示例文件[employee.json](http://drill.apache.org/docs/querying-json-files/)包含一些虚构的员工数据，您可以使用自定义函数进行查询。

### 简单的函数示例

本示例myaddints在employee.json文件的查询中使用简单的函数 。

如果您向Drill发出以下查询，则可以看到文件中的所有员工数据employee.json：

0: jdbc:drill:zk=local> SELECT \* FROM cp.`employee.json`;

该查询返回以下结果：

| employee\_id | full\_name | first\_name | last\_name | position\_id | position\_title | store\_id | department\_id | birth\_da |

+-------------+------------+------------+------------+-------------+----------------+------------+---------------+----------+-----------

| 1101 | Steve Eurich | Steve | Eurich | 16 | Store Temporary Checker | 12 | 16 |

| 1102 | Mary Pierson | Mary | Pierson | 16 | Store Temporary Checker | 12 | 16 |

| 1103 | Leo Jones | Leo | Jones | 16 | Store Temporary Checker | 12 | 16 |

…

由于postion\_id和store\_id列包含整数，所以可以myaddints在这些列上使用自定义函数发出查询，以在列中添加整数。

以下查询告诉Drill将myaddints函数应用于 文件中的列position\_id和store\_id列employee.json：

0: jdbc:drill:zk=local> SELECT myaddints(CAST(position\_id AS int),CAST(store\_id AS int)) FROM cp.`employee.json`;

由于JSON文件不存储关于数据类型的信息，因此必须CAST在查询中应用该 函数以告知Drill列中包含整数值。

该查询返回以下结果：

+------------+

| EXPR$0 |

+------------+

| 28 |

| 28 |

| 36 |

+------------+

…

## 自定义函数接口

实现适合您要开发的函数类型的Drill界面。每个接口都提供了一组需要的持有者，您可以在其中输入您的函数使用的数据类型，以及Drill调用的所需方法来执行您的函数的操作。

### 简单的功能界面

当你开发一个简单的功能，你实现了DrillSimpleFunc接口。函数的名称由您分配给name变量的字符决定。例如，以下简单功能的名称是myaddints：

@FunctionTemplate(name = "myaddints", scope = FunctionScope.SIMPLE, nulls = NullHandling.NULL\_IF\_NULL)

public static class Add1 implements DrillSimpleFunc{

该nulls = NullHandling.NULL\_IF\_NULL变量告诉Drill将NULL值作为NULL返回。对于大多数情况下，这个设置就足够了。如果您想更改Drill处理NULL值的方式，可以将设置更改为nulls = NullHandling.INTERNAL。

简单的功能界面包括你@Param和@Output你的功能可以处理的数据类型。

#### @Param持有人

该持有者指示函数作为输入处理的数据类型，并确定您的函数在查询中接受的参数数量。例如：

@Param BigIntHolder input1;

@Param BigIntHolder input2;

#### @输出持有人

该持有者指示处理返回的数据类型。例如：

@Output BigIntHolder out;

#### 设置和评估功能

简单的函数接口还包括两个必需的方法，setup和eval，Dri​​ll在处理查询时调用函数。

* 设置功能执行初始化和钻取仅执行一次的处理。
* eval函数包含的代码告诉Drill在数据列上执行什么操作。您将自定义代码添加到此方法。

#### 例

以下示例显示为该myaddints函数创建的程序：

package org.apache.drill.udfs;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

import org.apache.drill.exec.expr.DrillAggFunc;

import org.apache.drill.exec.expr.DrillSimpleFunc;

import org.apache.drill.exec.expr.annotations.FunctionTemplate;

import org.apache.drill.exec.expr.annotations.FunctionTemplate.FunctionScope;

import org.apache.drill.exec.expr.annotations.FunctionTemplate.NullHandling;

import org.apache.drill.exec.expr.annotations.Output;

import org.apache.drill.exec.expr.annotations.Param;

import org.apache.drill.exec.expr.annotations.Workspace;

import org.apache.drill.exec.expr.holders.BigIntHolder;

import org.apache.drill.exec.expr.holders.Float8Holder;

import org.apache.drill.exec.expr.holders.IntHolder;

import org.apache.drill.exec.expr.holders.VarCharHolder;

public class MyUdfs {

@FunctionTemplate(name = "myaddints", scope = FunctionScope.SIMPLE, nulls = NullHandling.NULL\_IF\_NULL)

public static class Add1 implements DrillSimpleFunc{

@Param BigIntHolder input1;

@Param BigIntHolder input2;

@Output BigIntHolder out;

public void setup(){}

public void eval(){

out.value = input1.value + input2.value;

}

}

### 聚合函数接口

当你开发一个聚合函数时，你实现了这个DrillAggFunc接口。函数的名称由您分配给name变量的字符决定。例如，以下集合函数的名称是mysecondmin：

@FunctionTemplate(name = "mysecondmin", scope = FunctionTemplate.FunctionScope.POINT\_AGGREGATE)

public static class MySecondMin implements DrillAggFunc {

集合函数接口包含指示您的函数可以处理的数据类型的持有者。这个接口包括前面描述的@Param和@Output持有者，还包括@Workspace持有者。

#### @Workspace持有人

该持有者指示用于在处理期间存储中间数据的数据类型。例如：

@Workspace BigIntHolder min;

@Workspace BigIntHolder secondMin;

集合函数接口还包含在处理具有函数的查询时Drill调用的以下方法。

* 设置执行Drill仅执行一次的初始化和处理。
* 添加进程每个记录。它将函数应用于Drill处理的列中的每个值。
* 输出返回聚合函数的最终结果; 由Add方法应用的处理的计算值。这是Drill调用的最后一个方法。Drill在处理完所有记录后调用一次。
* 重置您提供此方法中的代码，用于确定在列中的数据类型从一种类型更改为另一种类型时（例如从int到float），Drill所采取的操作。在处理无模式数据之前，Drill扫描数据并隐式地尝试识别与每列相关联的数据类型。如果Drill无法识别与每列相关联的模式，则Drill处理列，假定该列包含特定的数据类型。如果Drill在列中遇到另一个数据类型，则Drill调用reset方法来确定如何处理场景。

#### 例

以下示例显示为该mysecondmin函数创建的程序：

package org.apache.drill.udfs;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

import org.apache.drill.exec.expr.DrillAggFunc;

import org.apache.drill.exec.expr.DrillSimpleFunc;

import org.apache.drill.exec.expr.annotations.FunctionTemplate;

import org.apache.drill.exec.expr.annotations.FunctionTemplate.FunctionScope;

import org.apache.drill.exec.expr.annotations.FunctionTemplate.NullHandling;

import org.apache.drill.exec.expr.annotations.Output;

import org.apache.drill.exec.expr.annotations.Param;

import org.apache.drill.exec.expr.annotations.Workspace;

import org.apache.drill.exec.expr.holders.BigIntHolder;

import org.apache.drill.exec.expr.holders.Float8Holder;

import org.apache.drill.exec.expr.holders.IntHolder;

import org.apache.drill.exec.expr.holders.VarCharHolder;

public class MyUdfs {

@FunctionTemplate(name = "mysecondmin", scope = FunctionTemplate.FunctionScope.POINT\_AGGREGATE)

public static class MySecondMin implements DrillAggFunc {

@Param BigIntHolder in;

@Workspace BigIntHolder min;

@Workspace BigIntHolder secondMin;

@Output BigIntHolder out;

public void setup() {

min = new BigIntHolder();

secondMin = new BigIntHolder();

min.value = 999999999;

secondMin.value = 999999999;

}

@Override

public void add() {

if (in.value < min.value) {

min.value = in.value;

secondMin.value = min.value;

}

}

@Override

public void output() {

out.value = secondMin.value;

}

@Override

public void reset() {

min.value = 0;

secondMin.value = 0;

}